Titre: Title:	Architecture pair-à-pair de grille d'ordinateurs de prochaine génération basée sur les services et les contraintes de qualité de service
Auteur: Author:	Ibrahim Georges Zreik
Date:	2010
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Zreik, I. G. (2010). Architecture pair-à-pair de grille d'ordinateurs de prochaine génération basée sur les services et les contraintes de qualité de service [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/250/

UNIVERSITÉ DE MONTRÉAL

ARCHITECTURE PAIR-À-PAIR DE GRILLE D'ORDINATEURS DE PROCHAINE GÉNÉRATION BASÉE SUR LES SERVICES ET LES CONTRAINTES DE QUALITÉ DE SERVICE

IBRAHIM GEORGES ZREIK

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INFORMATIQUE)

FÉVRIER 2010

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ARCHITECTURE PAIR-À-PAIR DE GRILLE D'ORDINATEURS DE PROCHAINE GÉNÉRATION BASÉE SUR LES SERVICES ET LES CONTRAINTES DE QUALITÉ DE SERVICE

présenté par : ZREIK Ibrahim Georges

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

Mme BOUCHENEB Hanifa, Doctorat., présidente

M. PIERRE Samuel, Ph.D., membre et directeur de recherche

M. ANTONIOL Giuliano, Ph.D., membre

DÉDICACE

À ma Famille, sans laquelle tout cela
n'aurait pas été possible.

REMERCIEMENTS

Je tiens à remercier le professeur Samuel PIERRE, pour l'intérêt qu'il a porté à mon sujet de recherche ainsi que pour ses conseils et ses encouragements. De même, je tiens à remercier les responsables ainsi que les membres du laboratoire LARIM que je côtoie depuis plus de 4 ans maintenant, et qui m'ont accompagné tout au long de mon apprentissage à l'École Polytechnique de Montréal.

Finalement, je tiens à remercier aussi les membres de ma famille et amis proches qui m'ont aidé à clarifier et réviser ce mémoire.

RÉSUMÉ

Grâce aux grilles informatiques (GRID), les projets difficiles à résoudre sont devenus envisageables pour les communautés scientifiques de petite et moyenne taille : des projets tels que des simulations d'écosystèmes, des simulations météorologiques, d'analyses de molécules, d'analyses de signaux, de recherches biomédicales ou autres problèmes ayant de réels impacts socio-économiques. Favorisant ainsi les découvertes scientifiques, les GRID depuis le début des années 90 ont vu leur popularité s'accroître. Aujourd'hui, elles ne se limitent plus qu'aux applications scientifiques, nous les retrouvons dans les plateformes de commerce électronique et de services Web. Leurs utilisations dépassent dorénavant les limites traditionnelles d'un réseau privé et il devient courant de voir des GRID faisant appel à des ressources tierces d'autres réseaux. Cependant, cette expansion des GRID à d'autres réseaux ne garantit plus une topologie fixe des ressources. L'environnement d'utilisation des GRID devient très souvent dynamique ou même mobile et les GRID existantes ne nous permettaient pas d'exploiter les ressources mobiles efficacement. Dans un environnement où il y a fréquemment des déconnexions entre les liens des ressources de la GRID, le retard dans le traitement d'une requête augmente significativement. La problématique de la faible performance des GRID de troisième génération dans des environnements dynamiques et mobiles nous a encouragé à fixer des objectifs d'amélioration de l'architecture des GRID.

L'architecture GRID proposée dans ce mémoire introduit une nouvelle génération de GRID plus flexible, plus dynamique, mais aussi intelligente. En effet, les nouveaux concepts de mobilité introduits dans l'architecture de notre GRID lui permettent de s'adapter aux environnements dynamiques, mais aussi de prendre en considération la mobilité de certaines ressources et la qualité de service des réseaux utilisés. Pour y arriver, dans une première phase, nous avons amélioré le support de la mobilité des tâches et des ressources des GRID existantes. Suite à cela, nous y avons introduit des concepts de mobilité : l'habilité de transférer à l'avancement des tâches à d'autres ressources de la GRID; l'habilité de poursuivre le traitement des tâches déjà en cours sans nécessairement être connecté en permanence à la GRID; et pour finir, l'habilité de coopérer directement avec d'autres ressources. De plus, la *topologie* de la GRID a aussi été révisée afin de marquer le changement de génération de GRID. Dans un groupe de travail d'une GRID, les ressources ont été rassemblées en *grappes* hiérarchisés par un ensemble de *têtes de grappe*. Le nombre de *têtes de grappe* dans un groupe de travail et de

ressources formant une *grappe* n'est pas fixe, la topologie se base sur des indices de performances et de qualité de services pour s'adapter aux environnements des différentes ressources de la GRID.

Afin de tester notre proposition, nous avons développé un simulateur dans lequel nous avons implémenté le comportement général des différentes générations de GRID, dont celle que nous proposons. L'évaluation de notre architecture GRID démontre que notre GRID de nouvelles générations se comporte aussi bien que les GRID existantes dans un environnement stable, peu dynamique et en ne considérant que des ressources voisines. Il est en effet difficile de dépasser les performances des GRID déjà existantes dans des environnements locaux. La qualité de service de ces réseaux y étant fixe et élevée, les nouveaux concepts de mobilités introduits dans notre architecture GRID n'améliorent pas le fonctionnement global de notre GRID dans ce type d'environnement.

L'amélioration des performances de notre architecture GRID par rapport aux autres générations de GRID sont visibles dès que l'environnement devient plus dynamique et que nous considérons des ressources distantes. La qualité de service des liens entre des nœuds de différents réseaux est rarement idéale. Pour communiquer entre deux ressources, les messages ne passent plus par un ou deux câbles d'une même technologie, mais par un ensemble de chemins possibles de différente qualité de service et de technologie. L'amélioration des performances est encore plus significative lorsque le nombre de têtes de grappe augmente ainsi que l'indice de mobilité des nœuds. Par exemple, en considérant que 80 % des ressources de la GRID sont dans un environnement dynamique, nous améliorons la performance des GRID existantes de 48 % dans les meilleurs cas mesurés.

ABSTRACT

Thanks to the GRID computing, complex projects became possible to be solved by scientific communities of small and medium size. Projects such as ecosystems simulations, metrological simulations, molecules analyses, signals analyses, biomedical research or other problems having real socio-economic impacts. Thus supporting the scientific discoveries, the GRID since the beginning of the Nineties saw their popularities increasing. Today, they aren't anymore limiting by scientific applications. Now, we find them in business platform and also Web services. Their uses exceed the traditional limit of a private network. It becomes current to see GRID calling upon third-party resources through other networks and Internet. However, this expansion of the GRID to other networks doesn't guarantee a stable topology of the resources. The environment of use of the GRID becomes usually dynamic or even mobile.

The GRID architecture that we suggest introduces a new generation of more flexible, more dynamic and also intelligent GRID. Indeed, the new concepts of mobility introduced into the architecture of our GRID consider the mobility of certain resources and the quality of service of networks used. With it, GRIDs can now evolve dynamically with environments.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

1G	Première génération
2G	Deuxième génération
3G	Troisième génération
4G	Quatrième génération
CPU	Unité centrale d'une ressource informatique
GRID	Grille d'ordinateurs
HD	Disque dur d'une ressource informatique
MPI	Bibliothèque logicielle de calcul parallèle (*Message Passing Interface)*
OpenMP	Bibliothèque logicielle de calcul parallèle (*Open Multi-Processing*)
OS	Système d'exploitation (*Operating System*)
P2P	Pair à Pair
QdS	Qualité de service
RAM	Mémoire vive d'une ressource informatique
UML	Langage de modélisation unifié *(Unified Modeling Language)*
XML	Langage extensible de balisage *(Extensible Markup Language)*

LISTE DES ANNEXES

CHAPITRE 1
INTRODUCTION

De nos jours, les réseaux d'ordinateurs sont omniprésents que ce soit dans les milieux publics, les milieux privés ou mêmes chez-soi. Ces réseaux sont composés d'interconnexions qui mettent à notre disposition un grand nombre de machines pouvant partager leurs puissances de calcul, faisant d'eux des ressources très convoitées. Issue de cette idée de partager des ressources, un grand nombre de projets proposent des architectures de *grilles informatiques (GRID)* permettant d'exploiter les ressources inutilisées pour accélérer les performances d'exécution d'applications. Il devient donc envisageable d'accélérer le processus aux longues recherches de solution de problèmes dits *NP-difficiles* ou *NP-complets* sans nécessairement disposer d'une importante infrastructure informatique, telle qu'un superordinateur ou un ensemble de clusters. Grâces aux GRID, des projets difficiles à résoudre tels que les simulations d'écosystèmes, les simulations météorologiques, l'analyses de molécules, l'analyses de signaux, les recherches biomédicales ou autres problèmes avec un réel impact socio-économique, sont devenus des projets envisageables pour des communautés scientifiques de petite et moyenne taille. Favorisant ainsi les découvertes scientifiques, les GRID, depuis le début des années 90 ont vu leur popularité s'accroître. Aujourd'hui, elles ne se limitent plus qu'aux applications scientifiques. Nous les retrouvons dans des plateformes de commerces électroniques ainsi que dans des plateformes de *services web*. Leurs utilisations dépassent dorénavant les limites traditionnelles d'un réseau privé. Il devient courant de voir des GRID utilisant des ressources tierces à travers divers réseaux tels que l'Internet. Cependant, cette expansion des GRID à d'autres réseaux ne garantit plus une *topologie* des ressources fixes. La *topologie* étant en constante évolution, l'environnement d'utilisation des GRID passe donc d'un état très souvent stable à un *état dynamique* ou même *mobile*.

Dans ce chapitre d'introduction, nous présenterons tout d'abord les concepts de base liés au domaine des GRID et des réseaux informatiques, qui sera suivi de la présentation des éléments de la problématique soulevés par notre recherche. Suite à cela, nous exposerons les objectifs et les activités envisagées pour les réaliser. Nous terminerons par un aperçu du plan de notre mémoire.

1.1 Définitions et concepts de base

Une *ressource* est une machine munie d'au moins un processeur (CPU) pouvant réaliser des calculs. Les plus répandues sont les ordinateurs de bureau et les ordinateurs portables, ainsi que les PDA, téléphones cellulaires, de mêmes que les consoles de jeux vidéo. Leurs principales caractéristiques sont *leurs puissances de calcul* et leurs espaces mémoire.

La *puissance de calcul* est grossièrement quantifiée comme la somme des fréquences des processeurs composant une ressource. Plus la fréquence est élevée, plus la puissance de calcul l'est aussi. La puissance de calcul représente le nombre d'instruction informatique pouvant être traitées à la seconde par la ressource. En général, un nœud ayant une puissance de calcul de 1 GHz peut traiter 10^9 instructions informatiques en une seconde.

Un *superordinateur* est un ordinateur composé d'un très grand nombre de processeurs mis en parallèle. La particularité de cette ressource informatique est qu'elle offre une énorme puissance de calcul dans un seul et même ordinateur.

Un *cluster*, aussi appelée une « grappe de serveurs », est un ensemble de ressources homogènes dédiées, localisées dans un même environnement et organisées de manière centralisée. Elles sont gérées de manière globale et permettent de dépasser les limites de calcul des ressources actuelles.

Une *grappe* est un regroupement de ressources indépendantes et hiérarchisées. Leurs dispositions dans le réseau suivent une topologie centralisée autour d'une ressource coordinatrice surnommée la « tête de grappe ».

Une *grille informatique*, aussi appelée GRID, est définie comme étant un système informatique parallèle et distribué. Son objectif est de traiter les requêtes qu'il reçoit. Elle permet d'utiliser plus efficacement des ressources informatiques disponibles dans une entreprise, un laboratoire ou même sur l'Internet. Cela, sans affecter la productivité des utilisateurs avec lesquels elle partage ces ressources. Les principales caractéristiques des GRID sont de supporter un très grand nombre de ressources hétérogènes et distribuées géographiquement de manière souvent dynamique et d'offrir de hautes performances en termes de puissances de calcul. Tous ces calculs, se font de manière virtuelle et transparente. La GRID regroupe des ressources ayant différentes puissances de calculs, différentes tailles de mémoires virtuelles, assemblés par différents constructeurs et exécutant différent *systèmes d'exploitation* (OS).

Une *requête* est une tâche globale à réaliser par une GRID. Une requête doit être divisible en plusieurs tâches. Chaque tâche sera traitée par une ressource ou divisée en sous-tâches qui seront redistribuées à d'autres ressources.

Une *topologie* de réseau représente *l'architecture d'un réseau*. Elle informe sur le modèle d'organisation du réseau. Elle détermine la position de chacune des ressources informatiques (surnommé *nœud*) dans le réseau ainsi que les liens existants qui les relie.

Un *nœud* représente une ressource informatique dans un réseau. C'est un ensemble de nœuds interconnectés qui forme la topologie d'une GRID.

Un nœud *voisin* est un nœud avec lequel il existe une connexion réseau direct pour communiquer avec lui.

Une *table de routage* est une table d'information qui contient les chemins des nœuds par lesquels il faut passer pour envoyer un message à un autre nœud du réseau. Si le nœud destinataire est un nœud voisin, le chemin dans la table de routage pour le joindre sera vide puisque les nœuds peuvent communiquer directement. Sinon, le chemin sera composé de la suite des nœuds intermédiaires par lesquels les messages devront passer afin d'arriver au nœud destinataire.

Une topologie *centralisée* est un modèle en réseau informatique. Elle est composée d'un « serveur » s'occupant de la coordination des traitements des requêtes, et d'un ensemble de « clients » connecté au serveur qui traite les tâches qui lui sont assignées par celui-ci. Nous parlons de réseau centralisé, car toutes les communications et échanges d'information entre clients passent par l'intermédiaire du serveur.

Une topologie *paire à paire* surnommée P2P, est un modèle en réseau informatique dont les ressources peuvent autant être « serveur » que « client » ou même les deux. Ce modèle s'oppose à une topologie centralisée, car il n'est plus nécessaire pour un client de passer par un serveur pour communiquer avec un autre client. Il est souvent formé de ressources indépendantes et anonymes.

Un environnement dit d*ynamique* est un réseau en constante évolution dans le temps. Cette dynamique force aussi la topologie des ressources informatiques hébergées dans le réseau à évoluer constamment avec elle. De nouvelles ressources peuvent s'ajouter à tout moment au réseau, mais d'autres peuvent aussi le quitter. Une déconnexion au réseau peut être volontaire ou involontaire de la part d'une ressource. Si elle est involontaire, nous parlerons de déconnexion forcée, sinon nous parlerons plutôt d'une déconnexion douce.

Un environnement dit *mobile* est un environnement très dynamique dans lequel les ressources peuvent se déplacer géographiquement, tout en restant connectées et accessibles. Généralement, la liberté de mouvement de ces éléments est limitée par les contraintes de la qualité de service des liens.

La *qualité de service* d'un réseau (QdS) est la capacité de transmettre du trafic sur le réseau en garantissant certaines contraintes de qualité comme le débit, la bande passante, le délai de transmission, le taux de perte de paquets, etc.

Un *service web* est un intergiciel de communication permettant l'échange de donnée entre différents systèmes et applications.

Un problème dit *NP-difficile* ou *NP-complet* est un problème pour lequel à partir d'une certaine taille, il n'existe aucun algorithme polynomial pouvant le résoudre. Pour cela, nous utilisons des méthodes de résolution appelée *métaheuristiques*.

Les *métaheuristiques* sont un ensemble de méthodes d'optimisation et de résolution algorithmiques permettant de faire une approximation de l'optimum global d'un problème à résoudre en se rapprochant rapidement de la meilleure solution possible.

L'accélération parallèle $\propto$ est une métrique permettant d'évaluer la performance d'une parallélisations. En supposant que tous les processeurs ont la même puissance de calcul, ce facteur d'accélération consiste à faire le rapport entre le temps d'exécution séquentielle t_s et le temps d'exécution parallèle t_p sur l'ensemble des n processeurs disponibles :

$$\propto = \frac{t_s}{t_p(n)} \tag{1.1}$$

Théoriquement, l'accélération parallèle idéale est atteinte quand $\propto = n$. Dans cette situation idéale, le temps de calcul parallèle t_p est égal au temps de calcul séquentiel t_s divisé par le nombre de processeurs n.

Un *agent mobile* est une entité logicielle qui se déplace de plateforme (ressource informatique) en plateforme. Il représente généralement une tâche à effectuer. Il se charge de négocier avec chaque plateforme sur son chemin afin de s'assurer qu'elle l'aidera à la réalisation de la tâche qu'il accompagne. Une fois sa tâche complétée, il retourne à la plateforme qui l'a envoyé.

1.2 Éléments de la problématique

Les GRID actuels sont très efficaces. Leurs architectures supportent un grand nombre d'applications et leurs flexibilités permettent d'exploiter un très grand nombre de ressources. Cependant, leur conception est orientée vers la performance ce qui ne leur permet pas de s'étendre au-delà des réseaux fixes. Même si leur popularité tend à les introduire dans des environnements dynamiques ou même mobiles, leurs performances s'y dégradent de manière significative.

En effet, la majorité des concepteurs de GRID ne prennent pas en considération la qualité de service des réseaux. Même s'il existe des mécanismes de qualité de service qui permettent aux requêtes de s'assurer que la GRID dispose des ressources nécessaires pour les exécuter, un grand nombre de facteurs ne sont pas pris en compte, tels que les délais de communication entre les ressources de la GRID, leur stabilité ainsi que leurs périodes de disponibilité. Présentement, une ressource d'un GRID ayant une tâche à réaliser se doit de la finir, sinon le travail effectué est perdu. Elle ne peut pas transmettre le travail déjà accompli à une autre ressource si elle désire se retirer de la GRID. D'autre part, même si l'architecture des GRID actuels est plus flexible, certains de ces éléments restent encore statiques, comme le choix de serveur de coordination. Toutes ces lacunes rendent l'utilisation des GRID actuels impossibles dans un environnement où les liens de communication sont souvent éphémères.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de proposer une architecture de GRID de nouvelle génération utilisable dans un environnement dynamique ou mobile. Cette nouvelle architecture pourra supporter de nouvelles applications mobiles et pourra optimiser l'utilisation des ressources mobiles.

Pour ne pas avoir à concevoir une architecture GRID de zéro, nous tenons pour acquis certains éléments ayant déjà fait leur preuve dans l'architecture des GRID actuelles, puis nous proposons d'améliorer les points suivants :

- La mobilité des ressources : Afin de rendre une GRID complètement dynamique et fonctionnelle dans un environnement mobile.
- La mobilité des tâches : Afin de décentraliser les groupes de travail et de rendre les ressources plus indépendantes dans le traitement de leurs tâches.

Un fois les améliorations apportées, nous compléterons notre architecture de GRID en introduisant les concepts suivants :

- le transfert de l'avancement des réalisations de tâches entre ressources;
- le travail « hors groupe » et « pair-à-pair ». Plus précisément, permettre à une ressource qui s'est déconnectée temporairement d'un groupe de travail, de finir sa tâche si possible seul ou avec d'autres ressources voisines. Puis d'envoyer ses résultats à la prochaine connexion au groupe;
- la formation en grappe des groupes de travail en considérant la qualité de service des réseaux exploités.

Pour ce faire, le cheminement de notre mémoire a été décomposé en trois grandes phases. Chaque phase constitue un ensemble d'activités à réaliser.

La première phase implique tout d'abord l'analyse de l'architecture des GRID précédentes. Avant de proposer une architecture de GRID, nous devons identifier clairement les différents éléments constituant les architectures de GRID des générations précédentes. Il nous faudra aussi déterminer l'impact d'une architecture GRID totalement dynamique sur les performances d'un réseau.

Dans la seconde phase, nous indiquerons les éléments des architectures GRID précédentes que nous prendrons pour acquis. Même si ces éléments sont déjà fonctionnels, nous allons proposer des améliorations afin de réduire les impacts de la mobilité des ressources sur la durée du traitement d'une requête. Nous remplacerons les éléments de l'architecture des GRID encore statiques par des éléments dynamiques. Pour organiser la topologie de notre GRID, nous allons présenter des métriques de mesures de la qualité de service et les mécanismes permettant d'optimiser les échanges de données au sein d'un groupe de travail.

Lors de la troisième phase, nous développerons l'architecture GRID que nous proposons. Nous utiliserons une plateforme indépendante aux systèmes d'exploitation des ressources. Le choix d'une plateforme indépendante pour développer notre architecture GRID nous permettra de regrouper un plus grand nombre de ressources hétérogènes. Puis nous testerons notre GRID dans différents types d'environnements et nous comparerons l'évolution de ces performances à celles des GRID précédentes.

1.4 Plan du mémoire

En suivant les différentes phases d'activités à réaliser présentées précédemment, le chapitre suivant portera sur un rappel des architectures GRID existantes. Nous présenterons et analyserons les différentes générations de GRID.

Au troisième chapitre, nous présenterons notre architecture GRID de nouvelle génération, les révisions apportées par rapport aux versions précédentes ainsi que les nouveaux concepts introduits tels, le transfert de tâche, le travail « hors groupe », le P2P et la formation en grappe du groupe de travail.

Les détails techniques de l'implémentation de la plateforme seront présentés au quatrième chapitre, mettant l'emphase sur les différents concepts et mécanismes propres à la nouvelle génération de GRID proposée. Nous réaliserons aussi une analyse des performances de notre architecture GRID comparativement aux générations précédentes.

En guise de conclusion, le cinquième chapitre fera un bref rappel des innovations introduites dans notre architecture GRID de nouvelle génération et proposera des améliorations futures.

CHAPITRE 2
ANALYSE DES ARCHITECTURES GRID

Le concept de GRID est l'aboutissement de plusieurs dizaines d'années de recherches et d'expériences avec un grand nombre de produits commerciaux. Pour mieux comprendre l'architecture des GRID, nous allons commencer dans ce chapitre par un survol de l'histoire des réseaux d'ordinateurs. Nous présenterons les différentes générations d'architecture des GRID actuels; pour chacune d'elle, nous exposerons leurs architectures globales, nous identifierons leurs avantages et inconvénients puis nous citerons quelques projets qui en ont découlé.

2.1 Bref historique des réseaux d'ordinateurs

Le besoin en puissance de calcul existe déjà depuis plus d'un demi-siècle. Il a fallu attendre 1965 pour que le premier ordinateur central IBM System/360 [1] voie le jour. À cette époque le mot « ordinateur » signifiait une machine de grande taille (pouvant s'étendre à toute une salle de réunion) dont le temps d'utilisation était partagé entre ses utilisateurs. Malgré la programmation des tâches à l'aide de cartes perforées, son imposante taille et sa faible puissance de calcul de 10^6 instructions informatiques par seconde (comparativement à 9,7 10^9 pour un processeur Intel Pentium 4 cadencé à 3,2 GHz) [2]; son fort succès a été l'initiateur des projets de recherche en informatique. Sa capacité à exécuter plusieurs programmes inspira le développement des ordinateurs des années suivantes.

C'est à partir des années 70 que les mini-ordinateurs ont été commercialisés. Légèrement moins performants, de la taille d'armoires et destinées au budget des moyennes entreprises, les mini-ordinateurs ont rapidement remplacé les ordinateurs centraux. Le souci de rentabiliser l'investissement dans ces machines coûteuses a poussé l'amélioration du système de partage de tâches. Contrairement aux ordinateurs centraux qui ne pouvaient exécuter qu'une tâche à la fois, le partage de la puissance de calcul a permis au mini-ordinateur d'être multitâche. Cette amélioration très attendue par la clientèle a été permise grâce à l'introduction du système d'exploitation multitâche tel qu'UNIX [3]. En effet, le système d'exploitation jouant le rôle d'intermédiaire entre les utilisateurs et le matériel, la programmation des tâches a été simplifiée et la planification de leurs exécutions a été automatisée. Malgré tous les avantages apportés par les mini-ordinateurs, leur essor n'a duré que quelques années.

L’avancement technologique des recherches en matériel informatiques a permis la réduction de tous les éléments de l’unité de calcul en un seul et unique petit ensemble de transistors. Cette prouesse nommée microprocesseur faite par la compagnie Intel à la fin des années 80 a introduit une troisième catégorie d’ordinateur, les « micro-ordinateurs » aussi connus sous le nom d’ordinateur personnel. Leurs faibles coûts ainsi que l’émergence des systèmes d’exploitation à interfaces graphiques tels que DOS et Windows, permirent au micro-ordinateur d’inonder les marchés informatiques jusqu'à aujourd’hui. Très rapidement, les ordinateurs centraux connus pour être la catégorie des ordinateurs les plus puissants ont été remplacés par des réseaux de micro-ordinateur, ouvrant ainsi la voie aux recherches dans le domaine des réseaux d’ordinateurs, soit au « Network Computing ».

Les clusters furent la première configuration réseau d’ordinateurs dédiée aux traitements massifs de tâches parallèles. La première version commerciale mature de cette technologie a vu le jour en 1984. Vendu à l’unité, chaque micro-ordinateur VAXcluster [4] pouvait réaliser 10^6 instructions informatiques par seconde. Leur particularité est qu’ils pouvaient être interconnectés entre eux pour ainsi former un ordinateur virtuel pouvant exploiter la puissance de calcul, partager les fichiers et les périphériques de tous les micro-ordinateurs interconnectés dans le même réseau local.

C’est en 1989 avec la première version du logiciel Parallel Virtual Machine (PVM) [5] que les regroupements d’ordinateurs dans un réseau local sont devenus populaires. C’est la première fois qu’il n’était plus nécessaire de disposer de superordinateur (descendant des ordinateurs centraux) pour traiter de lourdes tâches. Le logiciel PVM permettait à tout ordinateur pouvant communiquer sur le réseau avec le protocole TCP/IP [6] de former un superordinateur virtuel. L’explosion de l’Internet, l’amélioration des technologies de communication ainsi que l’émergence des échanges P2P ont amenés le concept de cluster à s’étendre au GRID. N’étant plus limitées à l’exploitation des ressources disponibles localement, des grilles de micro-ordinateurs hétérogènes dispersés géographiquement surnommés GRID ont permis de former des superordinateurs virtuels encore plus impressionnants. Un exemple du succès des GRID et le projet SETI@home [7] qui juste deux ans après son lancement en 1999 a regroupé plus de 3 millions de micro-ordinateurs, formant ainsi une puissance de calcul totale de 23,37 10^{12} opérations à virgule flottante par seconde.

De nos jours, les clusters, le P2P et les GRID sont les technologies couramment utilisées pour regrouper des micro-ordinateurs que ce soit pour :

- ✓ Améliorer la performance des micro-ordinateurs :
  - o Augmenter la puissance de calcul;
  - o Augmenter l'espace de stockage disque;
- ✓ Réduire le temps de traitement de tâches lourdes;
- ✓ Faciliter l'échange de données;
- ✓ Permettre la redondance des données;
- ✓ Accélérer la saisie et la recherche d'informations.

Bien que ces technologies permettent presque d'arriver aux mêmes fins, elles sont bien différentes. Pour mieux comprendre ces différents concepts, nous proposons les Tableaux comparatifs [8] 2.1 et 2.2 :

Tableau 2.1 - Tableau comparatif des réseaux d'ordinateurs (1re partie)

Caractéristiques	Cluster	GRID	P2P
Type de ressources	Micro-ordinateur de gamme moyenne	Micro-ordinateur de moyenne et haut de gamme	Micro-ordinateur de toutes gammes
Utilisateur	Multiple	Multiple	Multiple
Implémentation	Matériel et système d'exploitation	Intergiciel	Intergiciel
Découverte des ressources	Centralisée (local seulement)	Centralisée et décentralisée	Décentralisée
Administration	Centralisée	Centralisée et décentralisée	Décentralisée
Gestion des ressources	Centralisée	Centralisée et distribuée	Distribuée
Exécution des tâches	Centralisée	Centralisée et décentralisée	Décentralisée

Tableau 2.2 - Tableau comparatif des réseaux d'ordinateurs (2e partie)

Caractéristiques	Cluster	GRID	P2P
Intercompatibilité avec d'autres ressources	Ressources homogènes seulement (Non)	Ressources hétérogènes (Oui)	Ressources hétérogènes (Oui)
Une copie du système	Oui	Non	Non
Flexibilité (nombre de ressources du réseau)	100	1000 - Million	illimité
Performances	Fixe et garantie	Variable, mais haute	Variable
Taille des tâches	Moyenne	Lourde	Très lourde
Délai de réponse / Bande passante	Faible / Élevé	Élevé / Faible	Élevé / Faible

Les GRID sont définies comme étant un système informatique parallèle et distribué, permettant une utilisation plus efficace des ressources informatiques disponibles dans une entreprise, un laboratoire ou même sur l'Internet. Les tableaux comparatifs précédents mettent en évidence que les GRID sont les intermédiaires entre les clusters de faible population garantissant leurs performances et les réseaux P2P permettant une forte population de ressources décentralisées mais dont les performances globales ne sont pas garanties.

Depuis leur début, les GRID ont évolué jusqu'à aujourd'hui. Dans la suite de ce chapitre, nous présentons les différentes générations par lesquelles les GRID sont passées.

2.2 Générations des GRID

Le domaine de recherche sur les GRID est largement couvert par la littérature. Depuis la fin des années 90, ce domaine à débouché sur plusieurs projets commerciaux et scientifiques qui connaissent de grands succès tels : SETI@home, Entropia [9], XtremeWeb [10], Folding@home [11], Distributed.net [12] et toutes les autres architectures GRID qui sont basées sur l'idée d'exploiter les ressources inutilisées. Le plus populaire, SETI@home permet de profiter des

centaines de billions d'opérations à virgule flottante par seconde. Toute cette puissance de calcul est dédiée à l'analyse des signaux radio extraterrestres en exploitant en moyenne plus de 1,7 million d'ordinateurs. Pour clarifier l'évolution des GRID, Frank Capello [13] s'est proposé de catégoriser les GRID en trois générations selon l'architecture de l'intergiciel utilisé pour former la GRID et la topologie de leur interconnexion.

2.2.1 Première génération

L'architecture des GRID de *première génération* fut inspirée directement de celle des clusters. La topologie de l'architecture réseau des GRID de cette génération est centralisée tout comme celle des clusters. De ce fait, toutes les ressources formant la GRID ne sont interconnectées qu'à une ressource bien spécifique appelée « serveur de coordination ». La particularité de cette ressource est qu'elle est dédiée à la découverte de nouvelles ressources et à la soumission des tâches à traiter. C'est donc le serveur de coordination qui redistribue les tâches aux ressources formant la GRID. L'avantage de cette topologie centralisée est qu'elle permet de paralléliser massivement le traitement de tâche tout comme les clusters. Pour mieux représenter les interconnexions, la Figure 2.1 représente une GRID de première génération et ces différents éléments :

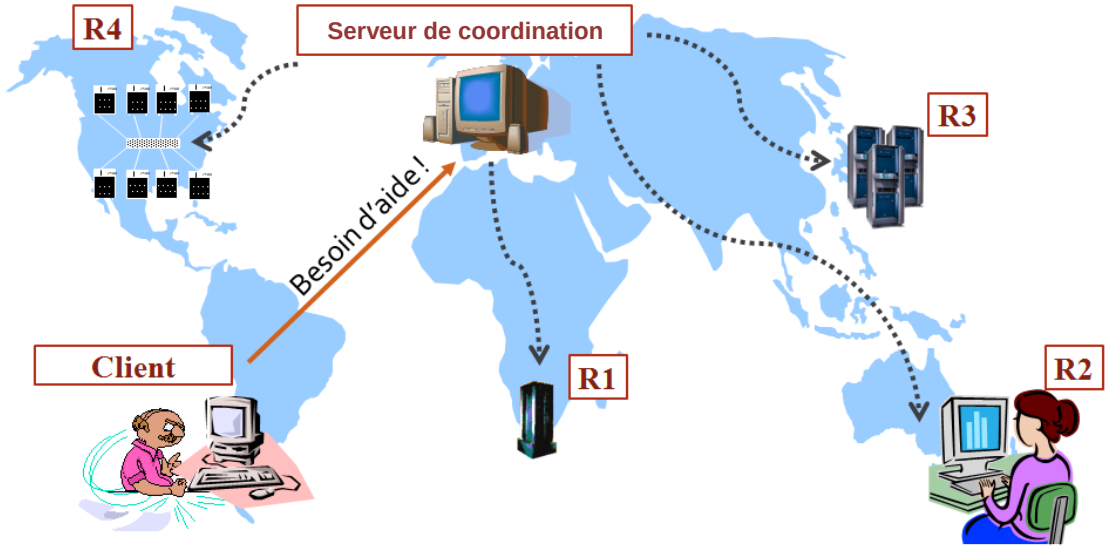


Figure 2.1 - Représentation d'une GRID de première génération

Dans cet exemple, un client autorisé se situant dans un laboratoire d'Amérique du sud à besoin d'aide pour traiter de lourdes tâches d'analyse; il soumet la tâche au serveur de

coordination situé en Europe. Le serveur de coordination sous-divise la lourde tâche en un ensemble de sous-tâches et les soumet aux ressources *R1*, *R2*, *R3* et *R4* faisant partie de sa GRID.

La différence entre les GRID de premières générations et les clusters réside dans l'implémentation des mécanismes permettant d'exploiter des ressources. Dans les GRID, les fonctions de partage de la puissance de calcul sont introduites dans une ressource informatique à l'aide d'un intergiciel plutôt que de les intégrer dans le matériel de la ressource. Grâce à cela, différents types de micro-ordinateurs de différents fabricants peuvent devenir des ressources d'un GRID tant qu'ils peuvent échanger des messages avec le protocole TCP/IP. Dans le domaine des clusters, les ressources doivent être compatibles entre elles au niveau de leurs logiciels et de leurs matériels, forçant ainsi à n'avoir que des ressources homogènes dans le réseau. De plus, les GRID des premières générations permettent de faire appel à des ressources distantes ne se trouvant ni dans le réseau local, ni géographiquement proche. Cette flexibilité permet aux GRID d'augmenter leurs populations de ressources considérablement. Cependant, plus le nombre de ressources éloignées est important dans la GRID, moins la performance de celle-ci est garantie.

Pour qu'une ressource puisse faire partie d'une GRID, il faut qu'elle possède l'intergiciel (adéquat et spécifique à la GRID dont il veut faire partie) installé dans son système d'exploitation. Il faut aussi qu'elle puisse communiquer avec le serveur de coordination, elle doit donc se trouver dans le réseau local de celui-ci ou pouvoir y accéder d'un autre réseau ou via Internet. La localisation du serveur de coordination de chaque GRID doit être connue à l'avance par les ressources pour pouvoir s'y connecter. Cette information est stockée dans l'intergiciel propre à chaque GRID.

Les projets gouvernementaux et scientifiques SETI@home et Folding@home cités précédemment sont tous deux des GRID de première génération. Leurs succès et le maintien de leur fonctionnement depuis déjà plus de 10 ans mettent en avant l'efficacité de ces réseaux d'ordinateurs GRID de premières générations. Cependant, leur principal désavantage est de ne pouvoir traiter qu'un type d'application. La GRID SETI@home est spécialisée dans le traitement de signal et celle de Folding@home dans l'analyse de molécule. Pour toutes autres nouvelles applications, il faudrait un nouveau projet pour définir une nouvelle GRID qui sera dédiée au traitement de chaque application. C'est cette limitation qui a motivé le changement à la génération suivante de GRID.

2.2.2 Deuxième génération

Motivés par le succès des GRID de première génération, plusieurs nouveaux projets dans le domaine des GRID ont vu le jour. Les plus connus sont Entropia, Alchemi [14], BOINC [15] et United Devices [16]. Contrairement aux GRID de première génération, ces projets supportent maintenant plusieurs types d'applications. Frank Capello les catégorise comme GRID de *seconde génération*. L'architecture générale de ces GRID reste inchangée, la soumission d'applications à traiter se fait par l'intermédiaire d'un serveur de coordination central qui redistribue les tâches aux ressources de sa GRID. De même, la topologie reste centralisée autour du serveur de coordination.

Une GRID n'étant plus conçue pour un problème bien spécifique, certaines entreprises offrent des services de partage de la puissance de calcul de leur GRID moyennant une rémunération. Au besoin, des entreprises ou laboratoires peuvent acheter une portion du temps de traitement de leurs tâches sur la GRID. Durant cette durée qui leur est allouée, toutes les ressources de la GRID se consacreront au traitement de cette tâche. Selon les GRID, il existe différents moyens de payement et modèle de partage des ressources des GRID. Pour mieux comprendre ce modèle d'affaire, nous proposons le schéma illustré à la Figure 2.2 :

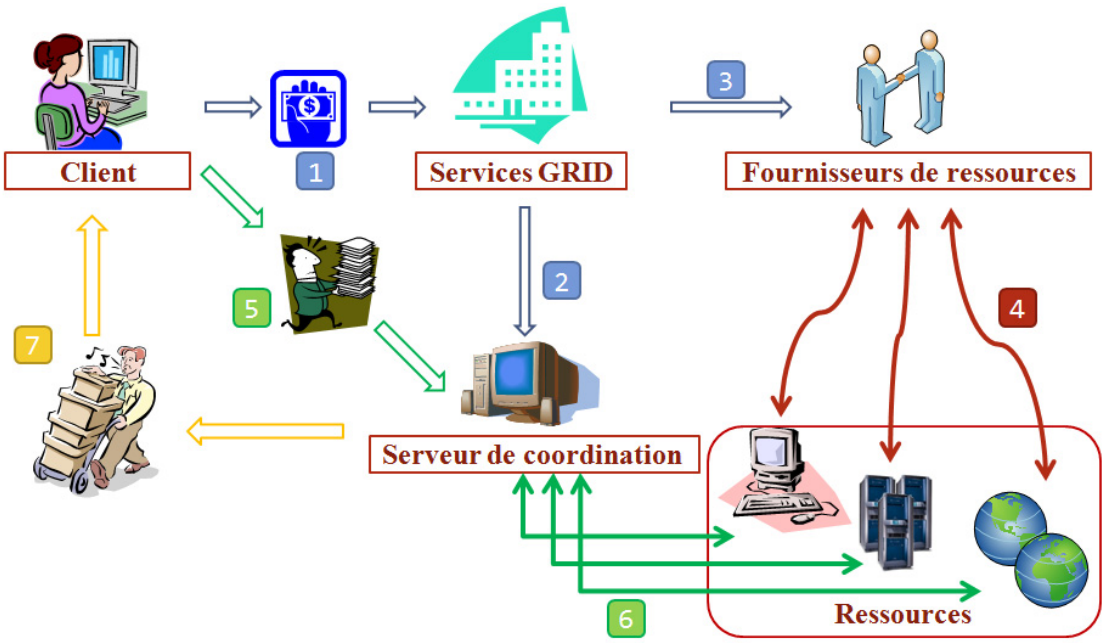


Figure 2.2 - Modèle d'affaire des GRID de deuxième génération

Tout commence par le besoin d'un client en puissance de calcul pour traiter une lourde tâche. Dans la première transaction (1), Il choisit un plan de service lui donnant accès à une GRID. Une fois le service payé, le client est autorisé à se connecter au serveur de coordination, (2). Une grosse partie du payement qui a été versé par le client aux services de GRID est donnée aux différents fournisseurs de ressources dans la transaction (3). Les concepteurs des GRID ne disposent pas eux-mêmes de million de micro-ordinateurs ! Ils exploitent des ressources mises en partage par des particuliers, des laboratoires et entreprises n'utilisant pas leurs ressources informatiques à plein temps, (4). Le client ayant maintenant accès au serveur de coordination, il lui soumet ces lourdes tâches à l'aide de la transaction (5). Le serveur de coordination sous-divise et distribue les tâches aux différentes ressources de sa GRID, (6). Une fois la tâche complète ou le temps alloué dépassé, les résultats correctement rassemblés et présentés retournent au client dans une transaction finale (7).

Ce modèle d'affaires a été possible grâce au support des langages de programmation parallèle qui ont été standardisés. Les GRID de deuxièmes générations supportent les librairies populaires d'*OpenMP* [17] et *MPI* [17]. Elles permettent donc aisément aux clients de programmer des tâches en *C* [18] pour la GRID.

Toutefois, les GRID de cette génération ne permettent pas la coopération des ressources sans passer par le serveur de coordination central. Cette limitation entraine une forte redondance des informations contenues dans les ressources et une surcharge du réseau de communication. Le serveur de coordination devient un goulot d'étranglement puisque tous les messages échangés entre les ressources passent par lui, ce qui entraine rapidement une baisse de performance des GRID. De plus, l'analyse [19] de certaines des GRID de seconde génération dénoncent la faible sécurité dans l'authentification des ressources et dans la validité des résultats qu'elles retournent.

Pour pallier le problème de redondance et d'accès aux informations, il a été proposé d'utiliser une mémoire tampon [20] entre les ressources. Néanmoins, cette méthode ne permet pas de réduire significativement la charge du réseau. La communauté scientifique s'est donc tournée vers le P2P afin de combler les lacunes des GRID de seconde génération ouvrant la porte aux GRID de *troisième génération*.

2.2.3 Troisième génération

Les GRID de troisième génération sont basées sur une architecture réseau plus flexible et dynamique faisant appel à des mécanismes de détection automatique de ressources et de sécurité

plus efficace. Plusieurs propositions d'architecture de GRID ont été faites dans la littérature à ce sujet. Nous en avons retenu trois. La première proposition fait appel à un regroupement de ressources avec plusieurs serveurs de coordinations [21]. Elle permet de réduire le goulot d'étranglement au niveau des serveurs de coordinations, mais n'apporte pas vraiment de nouvelles améliorations. La deuxième proposition ouvre la voie au P2P dans les GRID, BitDew [23] propose un réseau décentralisé de ressources totalement indépendantes. Chaque ressource de la GRID propose une liste de services de protocole d'échange P2P afin d'accélérer l'échange de données dans la GRID. Malgré ses idées innovatrices, BitDew se limite au transfert et au stockage de données, cette GRID ne propose pas de traitement de tâche. La troisième architecture de GRID retenue propose aussi une architecture réseau décentralisée P2P basée sur l'échange de services [22]; surnommée *Aneka* par ses concepteurs, cette architecture de GRID de troisième génération change la vision traditionnelle de l'architecture des intergiciels de GRID. Parmi les propositions précédentes, c'est cette troisième proposition qui nous a semblé être la plus pertinente.

La spécificité de la GRID Aneka est qu'elle utilise le même intergiciel pour tous les éléments de sa GRID. Il n'y a plus de différence entre un intergiciel de serveur de coordination et celui d'une simple ressource. Toutes les ressources sont désormais appelées « nœuds » de la GRID. C'est maintenant les services se trouvant dans l'intergiciel qui définit le rôle de chaque nœud. Suivant cette logique, les serveurs de coordinations des GRID de génération précédente deviennent des nœuds qui possèdent le service de coordination. Pour marquer cette différence, le titre de serveur de coordination est remplacé par celui d'« hôte ». Le schéma global de l'architecture de l'intergiciel d'un nœud peut être présenté comme le montre la Figure 2.3. Elle est divisée en quatre couches :

- Couche Réseau : Elle gère les connexions physiques avec les autres nœuds du réseau.
- Couche Sécurité : Elle s'assure que le nœud ne communique qu'avec d'autres nœuds autorisés et elle vérifie que les informations échangées sont autorisées.
- Couche Service : Elle regroupe l'ensemble des services hébergés par le nœud.
- Couche Ressource : Elle fait le pont entre l'intergiciel de la GRID et le système d'exploitation (OS) de la ressource. Elle collecte aussi périodiquement des informations sur le matériel de la ressource.

Leurs nœuds pouvant héberger différents services, les GRID de troisième génération ouvrent la voie à de nouveaux types d'application. Que ce soit pour le partage de fichiers, le besoin d'une base de données performante, le besoin de plateformes de services web, l'automatisation de transaction, la négociation d'échange de service (broker) et autres, les GRID ne sont plus limitées qu'au partage de la puissance de calcul de leurs ressources.

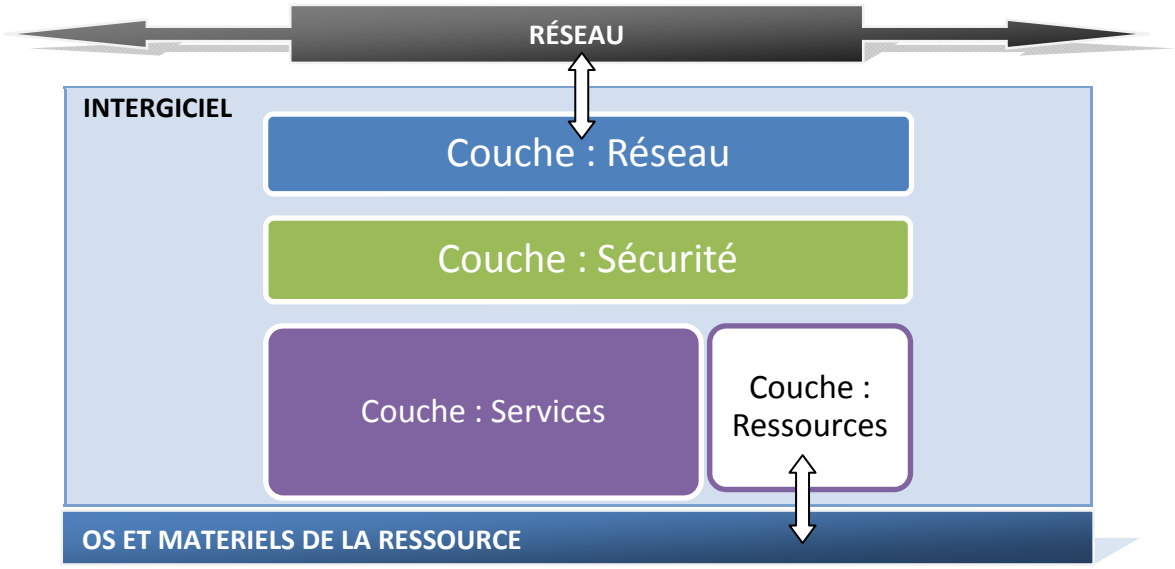


Figure 2.3 - Architecture d'un nœud de GRID de troisième génération

Leurs nœuds pouvant héberger différents services, les GRID de troisième génération ouvrent la voie à de nouveaux types d'application. Que ce soit pour le partage de fichiers, le besoin d'une base de données performante, le besoin de plateformes de services web, l'automatisation de transaction, la négociation d'échange de service (broker) et autres, les GRID ne sont plus limitées qu'au partage de la puissance de calcul de leurs ressources.

La GRID Aneka révise aussi l'architecture réseau en décentralisant la topologie des ressources. La découverte des nouveaux nœuds de la GRID se fait comme dans les réseaux P2P : chaque nœud signale sa présence dans le réseau où il se trouve et si d'autres nœuds y sont présents, alors ils s'associeront jusqu'à former une GRID de grande taille. Dans cette génération de GRID, il peut avoir autant de groupes de travail dans la GRID qu'il y a d'hôtes. Il n'est pas nécessaire qu'un client connaisse la position exacte des hôtes dans la GRID, il peut envoyer sa requête à n'importe quel nœud de celle-ci. Si le nœud ayant reçu la requête ne dispose pas du service de coordination, il retransmettra la requête à l'hôte de plus proche. Le traitement des tâches est quant à lui similaire à celui des GRID de première et deuxième génération, il se fait de

manière centralisée autour de l'hôte. Néanmoins, l'hôte peut ne pas faire appel à tous ces nœuds voisins pour traiter une requête. Le choix des nœuds qui feront partie du groupe de travail se fait à partir de l'équilibre entre les services qu'offrent les nœuds disponibles de la GRID et la qualité de service requise par la tâche à traiter.

Comme précisé dans la comparaison des différentes technologies existantes des réseaux d'ordinateurs présentée au Tableau 2.2 et contrairement au cluster, la performance des GRID est variable et non garantie. L'introduction de la notion de qualité de service pour le traitement des tâches a été rajoutée dans les GRID de troisième génération afin d'essayer de réduire la variation des performances et de garantir les performances nécessaires aux traitements des tâches.

Nous proposons le Tableau comparatif 2.3 afin de mieux identifier les différences entre les diverses générations de GRID :

Tableau 2.3 - Tableau comparatif des générations de GRID (1re partie)

Caractéristiques	1G	2G	3G
Administration	Centralisée	Centralisée	Centralisée et décentralisée
Découverte des ressources	Centralisée	Centralisée	P2P
Topologie	Centralisée	Centralisée	Centralisée et distribuée
Sécurité	Faible	Moyenne	Forte
Position exacte du serveur de coordination ou de l'hôte	Oui	Oui	Non
Garantie de performance	Non (variable)	Non (variable)	Faible (QdS de la requête)
Flexibilité (nombre de ressources)	1000 - Million	1000 - Million	illimité

Tableau 2.4 - Tableau comparatif des générations de GRID (2e partie)

Caractéristiques	1G	2G	3G
Une copie des sous-tâches	Oui	Dépendamment du projet de GRID	Non
Robustesse suite à la défaillance d'un nœud	Faible	Faible	Moyenne
Langage de programmation parallèle	Propriétaire	Standard	Standard, Services web
Projets	SETI@home, Folding@Home, Genome@Home	Entropia, Alchemi, BOINC, United Devices	XtremeWeb, BitDew, Aneka

Les propositions des GRID de troisièmes générations comblent les limitations générales des générations précédentes. En effet, la sécurité et la robustesse de la GRID ont été améliorées. Dans les GRID de troisième génération, nous retrouvons plusieurs copies des sous-tâches qui sont à réaliser. Cette redondance permet de remplacer un nœud défaillant sans faire échouer le traitement de la tâche en cours d'un client. Une autre amélioration non négligeable est l'ajout de mécanisme P2P pour la détection des nœuds de la GRID. Cette amélioration permet à la population des nœuds d'une GRID de s'étendre à tous les nœuds voisins de l'hôte sans limitation de taille. Nous pouvons aussi noter l'ajout des services web dans le langage de programmation parallèle qui permet au nœud de communiquer avec d'autres fournisseurs de service web ne faisant pas partie de la GRID tels que les banques et les agences de voyages, repoussant ainsi la limite de la nature des tâches pouvant être traitées par les GRID.

La garantie de performance reste faible même si certains efforts ont été réalisés à ce sujet. Dans la troisième génération de GRID, la qualité de service d'une requête a été introduite. Elle représente la puissance de calcul nécessaire à un groupe de travail pour pouvoir réaliser la requête dans un délai de temps raisonnable. Cependant, les concepteurs de ces GRID n'ont pas pris en considération la qualité de service du réseau. Le choix des chemins de communication ne considère pas les performances des liens de communication qui les composent. Les liens du réseau d'ordinateurs fonctionnent comme la topologie de la GRID devrait « être ». Le

raisonnement inverse serait nettement plus performant : la topologie de la GRID devrait suivre les liens recommandés du réseau d'ordinateurs. De plus, les périodes de disponibilité des ressources souvent partagées ne sont pas prises en compte. Une ressource ayant une tâche à réaliser se doit de la finir, sinon le travail effectué est perdu. Elle ne peut pas transmettre le travail déjà accompli à une autre ressource si elle désire se retirer du groupe de travail. D'autre part, même si l'architecture des GRID est plus flexible, certains de ces éléments restent encore statiques, comme le choix des hôtes.

Toutes les lacunes de cette dernière génération rendent l'utilisation des GRID impossible dans des environnements dynamiques et mobiles où les liens de communication sont souvent éphémères et le potentiel des ressources inutilisées non négligeable.

CHAPITRE 3
ARCHITECTURE GRID PROPOSÉE

Dans ce chapitre, nous présentons l'architecture GRID de nouvelle génération que nous proposons. Rappelons qu'une GRID est un regroupement de ressources dit nœuds collaborant entre eux afin de réaliser une tâche commune. Cette tâche commune est soumise par un client autorisé de la GRID sous forme d'une requête. La différence entre les générations de GRID réside dans l'architecture de leurs nœuds. Plus précisément, chaque intergiciel de nœud est composé d'un ensemble d'éléments ayant chacun un rôle bien précis à jouer. C'est le comportement global de ces éléments qui définit le fonctionnement de la GRID ainsi que sa topologie.

Comme nous l'avons annoncé dans le premier chapitre de notre mémoire, pour ne pas avoir à concevoir une architecture GRID de zéro, nous tenons pour acquis les mécanismes et outils suivants issus de l'architecture des nœuds des GRID de troisième génération basée sur les services :

- la détection dynamique des ressources;
- l'authentification des ressources et de la validité des résultats qu'elles retournent;
- l'échange de services entre ressources;
- la formation de groupes de travail pour traiter une requête;
- le langage informatique utilisé pour définir, envoyer, traiter et recevoir une requête à envoyer à une GRID.

Nous conservons aussi la division par couche des éléments de l'architecture d'un nœud d'une GRID de troisième génération avec services comme illustré à la Figure 2.3.

Tout d'abord, nous commencerons dans ce chapitre par identifier les éléments de l'architecture des nœuds déjà existants que nous utiliserons dans notre nouvelle architecture. Ces éléments étant déjà définis, nous ne ferons qu'un bref rappel de leurs rôles. Ensuite, nous présenterons les éléments que nous nous proposons de redéfinir afin de les améliorer. Puis, nous exposerons trois nouveaux concepts que nous introduirons aux architectures GRID de nouvelle génération.

3.1 Architecture héritée

Dans notre problématique initiale, nous avons soulevé le point que dans un environnement dynamique ou mobile, les architectures GRID existantes ne sont pas efficaces. Cependant, certains éléments présents dans les GRID de troisième génération ne changeront pas de fonctions dans l'architecture que nous proposons. Nous ne redéfinirons donc pas en détail ces éléments.

Dans cette section, nous allons identifier par couche, les éléments et mécanismes hérités de la troisième génération. Nous définirons brièvement leurs comportements globaux sans entrer dans les détails.

3.1.1 Couche : Réseau

Cette couche regroupe tous les éléments permettant l'interaction entre les nœuds. Nous allons présenter chacun des éléments de cette couche ainsi que leurs mécanismes.

3.1.1.1 Découverte des nœuds

La détection d'autres nœuds dans la GRID se fait de manière dynamique. Dès qu'un nouveau nœud se joint au réseau, il émet un signalement périodique dans celui-ci. Les nœuds formant déjà une GRID dans le réseau reçoivent son signalement et l'invitent à se joindre à la GRID. S'il n'y pas de GRID déjà existante dans le réseau, le nouveau nœud attendra l'arrivée d'un autre nœud pour en former une.

3.1.1.2 Groupes de travail

Quand une GRID de plusieurs nœuds est formée et qu'il y a une requête à traiter, un sous-groupe de nœuds de la GRID formera un groupe de travail. Ce sous-groupe peut être formé d'un seul nœud ou peut s'étendre à l'ensemble des nœuds de la GRID selon les besoins nécessaires pour effectuer le traitement de la tâche commune. Au niveau de la couche réseau, seuls les liens physiques avec les autres nœuds du groupe de travail seront considérés. Toutes les autres informations sur l'état du groupe de travail seront contenues dans les éléments de la catégorie « Information » (section 3.1.3.1) de la couche de services. Il est possible qu'un nœud fasse partie de plusieurs groupes de travail en même temps.

Les nœuds d'un groupe de travail pouvant être géographiquement éloignés, il n'est pas garanti qu'ils soient présents tout au long de la réalisation de leurs tâches. Dans le cas où un

nœud quitte le réseau, que ce soit à cause d'une déconnexion involontaire ou d'une déconnexion volontaire :

- la connexion avec ce nœud sera fermée;
- un signal sera émis à la couche de service afin de traiter ce changement.

Vice versa, si un nœud est ajouté au groupe de travail, cet élément pourra ajouter une nouvelle connexion à celles du groupe.

3.1.1.3 Passerelle d'échange de données

Une fois la connexion établie, un grand nombre d'informations sera échangé entre les différents nœuds. La passerelle d'échange de données transmettra aux bons nœuds les messages à envoyer et renverra les bonnes réponses à l'élément de l'architecture qui l'a envoyé.

3.1.2 Couche : Sécurité

Cette couche regroupe tous les éléments permettant l'authentification des ressources et la validité des résultats qu'elles retournent. Dans ce mémoire, nous ne nous penchons pas sur l'aspect de sécurité de la GRID. Cependant, nous allons brièvement présenter chacun des éléments de cette couche pour bien comprendre leurs rôles dans notre architecture.

3.1.2.1 Authentification

Cet élément permet à un nœud de vérifier l'identité des autres nœuds formant la GRID. Tous les nœuds identifiés pourront échanger des informations et des données avec celui-ci. Les autres nœuds non identifiés seront ignorés même s'ils tentent de communiquer.

3.1.2.2 Autorisation

Une fois un nœud identifié, il faut s'assurer qu'il ne puisse accéder qu'aux données pour lesquelles il a des droits d'accès. En effet, tous les nœuds n'ont pas le même rôle à jouer dans une GRID. Certains sont plus prioritaires que d'autres. Il n'est pas concevable que tous les nœuds accèdent à toutes les informations de tous les autres nœuds. Cet élément permet donc d'autoriser ou de refuser l'accès à certaines informations selon le privilège du nœud qui le demande. Un exemple simple serait qu'un nœud ne puisse pas demander à un autre nœud, ne faisant pas partie du même groupe de travail, des informations sur les tâches effectuées dans ce groupe.

3.1.2.3 Vérification

La dernière étape de la couche de sécurité est la vérification. Maintenant que les nœuds sont identifiés et autorisés à échanger, ce module s'assure que l'information échangée est valide. Entre autres, il s'assure que l'information n'est ni corrompue, ni altérée.

3.1.3 Couche : Services

Cette couche regroupe tous les éléments formant les services offerts par cette ressource aux autres nœuds. Le nombre de services offerts pouvant être nombreux, nous allons les regrouper en catégories afin de mieux les présenter.

3.1.3.1 Informations

Cette catégorie représente les éléments fournissant les informations reliées au nœud lui-même et à la GRID dont il fait partie. Nous retrouvons principalement trois services de catalogues :

- Catalogue des nœuds : Ce service répertorie tous les nœuds de la GRID. Nous y retrouvons leurs adresses IP, la liste des services qu'ils offrent ainsi que la spécification de leurs ressources matérielles.
- Catalogue des applications : Ce service liste tous les services et applications offerts par ce nœud. Ce catalogue est envoyé aux autres nœuds de la GRID.
- Catalogue des données : Un nœud peut se retrouver à traiter plusieurs tâches et possède plusieurs informations en même temps dans son entrepôt et sa cache. Ce service les indexe convenablement afin d'y avoir rapidement accès et de pouvoir gérer leurs différentes versions.

3.1.3.2 Entrepôt et cache

Dans cette catégorie, nous retrouvons toutes les données relatives aux tâches reçues et à l'avancement de leurs traitements. Pour réaliser correctement une tâche, il est nécessaire de définir quatre types d'information. Chacune d'elle doit être stockée de la réception au renvoi du résultat :

- Données relatives aux codes informatiques de la tâche à exécuter : Constitué de librairies, de codes déjà compilés ou même de code source selon les tâches à exécuter.

- Données relatives à la coopération avec les autres nœuds : Certaines tâches peuvent nécessiter une coopération avec d'autres nœuds durant leurs traitements. Ces données permettent de définir les interactions qui devront avoir lieu avec les autres nœuds du groupe de travail.
- Données relatives aux informations nécessaires à son exécution : Ces données en cache indiquent les valeurs des variables utilisées durant l'exécution de la tâche à laquelle ils sont associés.
- Données relatives à la présentation des résultats de l'exécution : Elle se présente sous forme d'un vecteur à remplir ou un fichier XML à compléter. Ces données représentent le formalisme dans lequel nous nous attendons à recevoir les résultats de la tâche qui sera exécutée sur ce nœud.

3.1.3.3 Gestionnaire de tâches

Les tâches reçues par le nœud et entreposées dans l'entrepôt doivent être exécutées. C'est cet élément qui assure leurs réalisations. Il peut être vu comme l'élément travailleur ou l'unité de calcul du nœud. Il effectue les opérations suivantes :

a) choisir la tâche la plus ancienne à exécuter se trouvant dans l'entrepôt;
b) valider ses requis;
c) exécuter le code informatique de la tâche;
d) rendre transparent l'accès aux informations associées se trouvant dans l'entrepôt et la cache;
e) rendre transparente la communication avec les autres nœuds du groupe de travail;
f) préparer les résultats de l'exécution dans le format demandé puis les retourner.

3.1.3.4 Stockages de données

Un nœud d'une GRID peut ne pas être seulement un réalisateur de tâches. Il peut aussi offrir des services de stockage tels qu'un serveur de fichier, une base de données, de réplication ou encore de mémoire tampon. Ces services sont secondaires, nous ne les détaillerons donc pas plus dans notre mémoire. Cependant, nous pouvons noter que ces types de services sont de plus en plus présents dans les GRID, ce qui rend l'utilisation des GRID encore plus attrayante.

3.1.3.5 Autres

Le concept de GRID basé sur l'échange de service rend très flexible les GRID. Nous pouvons y ajouter facilement de nouveaux services et de nouvelles applications. Nous n'avons présenté que les catégories les plus populaires mais il n'est pas rare de trouver d'autres types de services plus spécifiques selon les domaines dans lesquels nous utilisons la GRID. Un exemple d'autre service pouvant être disponible est l'interaction avec un système de payement pour charger l'utilisation d'un nœud. Les possibilités de nouveaux services sont donc infinies.

3.1.4 Couche : Ressources

Cette couche regroupe tous les éléments permettant l'interaction entre l'architecture du nœud et le matériel physique de celui-ci. Nous allons présenter chacun des éléments de cette couche.

3.1.4.1 Gestionnaire de l'unité de calcul

Cet élément évalue continuellement l'utilisation des CPU de la ressource. Selon leurs charges, il autorise ou interdit leurs partages à travers la GRID. Généralement, cet élément permet le partage des CPU s'il détecte qu'aucune session d'utilisateur n'a été ouverte sur la ressource. Dans le cas où une ou plusieurs sessions d'utilisateurs sont ouvertes, un autre mécanisme plus avancé de ce gestionnaire peut déterminer si les CPU sont peu utilisés pendant une certaine durée. Il peut alors autoriser leurs partages tant qu'il ne ralentira pas les processus dans les CPU déjà existants.

3.1.4.2 Gestionnaire de la mémoire virtuelle

Cet élément détermine la quantité de mémoire virtuelle restante et utilisable de la ressource. Cette quantité peut être évaluée dynamiquement ou fixée par l'administrateur du nœud.

3.1.4.3 Gestionnaire des disques durs

Cet élément détermine l'espace disque restant et utilisable de la ressource. Cette quantité peut être évaluée dynamiquement ou fixée par l'administrateur du nœud.

3.2 Les éléments révisés

Maintenant que nous avons identifié les éléments hérités de l'architecture des GRID de troisième génération, nous devons modifier certains d'entre eux afin d'atteindre les objectifs que nous nous sommes fixés. Entre autres, ces modifications ont pour but de changer légèrement les

comportements de ces éléments hérités afin de les adapter à un environnement dynamique et mobile.

Dans cette section, nous allons exposer les deux modifications de comportement à effectuer ainsi que les différentes couches et éléments affectés de l'architecture d'un nœud.

3.2.1 Support de la mobilité des nœuds

Dans les GRID de troisième génération, l'ajout et le retrait des nœuds se font de manière dynamique. À chaque déconnexion d'un nœud au GRID, celui-ci est retiré du groupe de travail et la tâche qu'il devait exécuter est redistribuée. Même s'il se reconnectait quelques secondes après, il ne ferait plus partie du groupe de travail. Au besoin, il peut y être réadmis, mais il sera considéré comme un nouveau nœud. Il recevra alors une nouvelle tâche.

Cette particularité des GRID de troisièmes générations défavorise énormément leurs performances dans un environnement dynamique ou mobile. Les interconnexions entre les nœuds de la GRID n'étant pas stables dans ces types d'environnements, plus le délai nécessaire à la GRID pour traiter une requête est long, plus le nombre de déconnexions est important. Afin d'estimer la perte de temps engendrée par ce mécanisme, nous allons tout d'abord modéliser le temps nécessaire à un groupe de travail pour traiter une requête. Pour simplifier notre modèle, nous supposons les points suivants :

- une requête est divisible en sous-tâche;
- un nœud ne peut traiter qu'une sous-tâche à la fois;
- il existe toujours un nœud dans le groupe de travail pouvant redistribuer les sous-tâches des nœuds déconnecté à d'autres nœuds;
- il existe toujours un nœud libre dans la GRID pour récupérer la sous-tâche d'un nœud qui s'est déconnecté;
- la réalisation d'une sous-tâche ne nécessite pas de coopération;
- tous les nœuds possèdent les mêmes caractéristiques physiques.

À l'aide de relation (3.1), nous pouvons déterminer le temps initial moyen nécessaire à la réalisation d'une requête soumise à un groupe de travail dans une GRID de troisième génération.

$$Ti_{Requete} = \frac{N_{SousTache} \times T_{SousTache}}{N_{noeud}} + P_{Deconnexion} \times N_{noeud} \times \left(\frac{T_{SousTachePerdu} + T_{Redistribution}}{N_{noeud}}\right) \quad (3.1)$$

Où :

- $Ti_{Requete}$ représente le temps initial moyen nécessaire pour traiter la requête;
- $T_{SousTache}$ représente le temps moyen nécessaire à un nœud pour réaliser une sous-tâche reçue;
- $N_{SousTache}$ représente le nombre de sous-tâches à traiter;
- N_{noeud} représente le nombre de nœuds faisant partie du groupe de travail;
- $P_{Deconnexion}$ représente la probabilité qu'un nœud se déconnecte durant le traitement de sa tâche;
- $T_{SousTachePerdu}$ représente le temps déjà alloué au traitement d'une sous-tâche à un nœud avant sa déconnexion. Il est borné par les valeurs suivantes :

  $$0 \leq T_{SousTachePerdu} < T_{SousTache}$$

  Il peut être remplacé par sa valeur moyenne $\frac{T_{SousTache}}{2}$, représentant une déconnexion durant le milieu du traitement ;
- $T_{Redistribution}$ représente le temps de synchronisation moyen nécessaire pour redistribuer une sous-tâche à un autre nœud de la GRID. Cette valeur peut être négligeable par rapport au temps de réalisation de la sous-tâche $T_{SousTache}$.

Dans une GRID de troisième génération, nous déduisons à l'aide de la relation (3.2) le temps initial moyen perdu durant le traitement d'une requête $Ti_{RequetePerdu}$:

$$Ti_{RequetePerdu} = P_{Deconnexion} \times N_{noeud} \times \left(\frac{T_{SousTachePerdu} + T_{Redistribution}}{N_{noeud}}\right) \quad (3.2)$$

Ou encore :

$$Ti_{RequetePerdu} = P_{Deconnexion} \times N_{noeud} \times \left(\frac{\frac{T_{SousTache}}{2} + T_{Redistribution}}{N_{noeud}}\right) \quad (3.2.1)$$

Les GRID étant utilisées généralement pour des tâches de longue durée, ce retard n'est pas négligeable, car il ne suffit que d'une déconnexion d'un nœud pour entraîner un retard d'en moyenne 50 % du délai de traitement normal de la requête par le groupe de travail. La relation (3.2) représente la progression moyenne du retard selon le nombre de nœuds qui se déconnectent. En réalité le retard répond à une fonction en escalier. Une seule déconnexion, entraîne un retard du groupe de travail de $T_{SousTachePerdu}$. Et cela, tant qu'il y a moins de nœuds qui se déconnectent que le nombre total de nœuds N_{noeud} dans le groupe de travail. S'il y a plus de déconnexion que ce nombre, le retard est alors doublé. S'il y a plus de déconnexion que le double de N_{noeud}, le retard est triplé et ainsi de suite.

Nous proposons donc de réduire $Ti_{RequetePerdu}$ dans le cas où le nœud déconnecté se reconnecterait dans un avenir proche. Pour ce faire, nous permettons à un nœud reconnecté de poursuivre le traitement qu'il avait commencé seulement s'il est toujours plus avancé que celui de sa copie redistribuée. La sous-tâche la moins avancée parmi les deux nœuds est donc arrêtée. Nous obtenons donc la révision du temps moyen perdu durant le traitement d'une requête $Tr_{RequetePerdu}$:

$$Tr_{RequetePerdu} = P_{Deconnexion} \times N_{noeud} \times \left(\left(\frac{T_{SousTachePerdu} + T_{Redistribution}}{N_{noeud}} \right) + P_{Reconnexion} \times \left(\frac{T_{DelaiReconnexion} - T_{SousTachePerdu} - T_{Redistribution}}{N_{noeud}} \right) \right) \quad (3.3)$$

Avec :

- $T_{DelaiReconnexion}$ représentant le délai moyen nécessaire au nœud pour se reconnecter;
- $P_{Reconnexion}$ représentant la probabilité qu'un nœud déconnecté puisse rejoindre le groupe de travail.

À l'aide des relations (3.2) et (3.3), nous faisons la déduction suivante :

$$Tr_{RequetePerdu} > Ti_{RequetePerdu} \Rightarrow T_{DelaiReconnexion} > T_{SousTachePerdu} + T_{Redistribution} \quad (3.4)$$

Avec $P_{Reconnexion} > 0$ et $N_{noeud} > 0$.

À partir de la relation précédente (3.4), dans le cas de la déconnexion d'un nœud et d'un $T_{Redistribution}$ négligeable :

- si $T_{DelaiReconnexion} \leq T_{SousTachePerdu}$, le délai supplémentaire ajouté au traitement de la requête par le groupe de travail sera équivalent à la durée de la déconnexion du nœud $T_{DelaiReconnexion}$;
- sinon, dans le cas où nous avons $T_{DelaiReconnexion} > T_{SousTachePerdu}$, le temps perdu est équivalent à celui d'une GRID de troisième génération $Ti_{RequetePerdu}$.

Ce mécanisme révisé sera plus adapté à un environnement mobile dans lequel les nœuds sont souvent indisponibles pendant une relève ratée entre deux points d'accès aux réseaux ou pendant le passage dans une zone non couverte. Cette révision permettra aussi aux nœuds dans un environnement dynamique d'être temporairement retirés du réseau sans engendrer de trop grandes pénalités. Pour ce faire, nous apportons des modifications à la couche réseau et de services.

3.2.1.1 Couche : Réseau

À la section 3.1.1, nous avons présenté trois éléments formant la couche réseaux de l'architecture d'un nœud. L'ajout du support de la mobilité des nœuds au niveau de la couche réseau nécessite seulement la modification de l'élément chargé de la gestion des groupes de travail. Suite à cette modification, il faut aussi adapter l'élément en charge de la distribution des données reçues et envoyées. Pour chacun de ces éléments, nous présenterons plus en détail leurs comportements révisés.

Groupes de travail

Dans une GRID de troisième génération, le rôle de cet élément est de mettre fin à la connexion avec un nœud s'étant déconnecté du réseau puis d'en avertir la couche de service. Nous améliorons ce comportement afin de rester à l'écoute d'un nœud déconnecté en vue d'une prochaine reconnexion. Une reconnexion sera permise si toutes les conditions suivantes sont satisfaites :

- ✓ l'*identifiant* (clef unique permettant d'identifier un nœud) est toujours le même;
- ✓ le nœud a effectué avec succès toutes les étapes de sécurité;
- ✓ la poursuite de traitement déjà entamé est pertinente.

Dans le cas d'une reconnexion d'un nœud, un signal de mise à jour sera envoyé à la passerelle.

Passerelle d'échange de données

Maintenant que nous permettons le retour d'un nœud dans son groupe de travail, nous devons aussi adapter cet élément. Désormais, s'il reçoit un signal de mise à jour, cet élément peut effectuer de manière transparente la redirection des messages au nœud reconnecté. Une fois la redirection effectuée, il émet un signal au catalogue des nœuds dans la couche de service pour retirer le nœud qui a servi de remplaçant.

3.2.1.2 Couche : Services

Le support de la mobilité des nœuds entraîne aussi la révision de deux éléments dans la couche des services : celui qui conserve les informations sur la topologie du groupe de travail ainsi que celui qui exécute les tâches reçues.

Informations

Dans la couche de service, l'élément d'informations contient un catalogue des nœuds du groupe de travail. Pour chacun des nœuds du groupe de travail, nous y retrouvons leurs adresses physiques et les services qu'ils offrent. Pour permettre correctement le retour d'un nœud déconnecté, il est nécessaire d'ajouter des informations supplémentaires à ce catalogue :

- L'identifiant du nœud : Clef unique permettant d'identifier un nœud de la GRID même s'il change d'adresse physique.
- L'Identifiant du nœud remplacé : L'ajout de cette information permettra de déterminer les nœuds déconnectés qui ont été remplacés par un autre nœud. Cela permettra un retour en arrière de la topologie initiale.
- L'état logique de la connexion : Information sur l'historique des données échangées. Ces informations permettront de déterminer si les informations dans un nœud reconnecté sont toujours compatibles avec celles des autres nœuds du groupe de travail.

Gestionnaire de tâches

Initialement, quand un nœud quitte son groupe de travail, le gestionnaire de tâches arrête la réalisation des tâches en cours et retire de l'entrepôt toutes celles qui sont liées à ce même groupe de travail. Si nous voulons permettre la mobilité des nœuds tout en conservant l'avancement de

leurs réalisations, il faut ajouter les comportements suivants dans le cas d'une déconnexion avec le groupe de travail :

- le traitement des tâches en cours est mis en veille. Elles sont alors stockées à nouveau dans l'entrepôt;
- après un décompte, si $T_{DelaiReconnexion} > T_{SousTachePerdu}$, le gestionnaire de tâches retire toutes les tâches de l'entrepôt liées à ce groupe de travail.

Toutes ses modifications ont permis aux nœuds d'un groupe de travail de se déconnecter temporairement du réseau, de changer de réseau et même d'adresse physique. Les nœuds peuvent maintenant être mobiles dans les réseaux physiques hébergeant la GRID. Comparativement aux GRID de troisième génération, ces mouvements n'engendrent que des retards équivalents ou moindres. Afin de compléter cette mobilité des nœuds, la deuxième série de modifications permettra aux tâches de l'être aussi.

3.2.2 Support de la mobilité des tâches

Dans les GRID de troisième génération, la soumission des tâches aux nœuds du groupe de travail se fait par l'intermédiaire d'un nœud administrateur appelé « hôte ». Il s'agit d'un nœud ayant le service de coordination dans sa couche de « services ». Il y en a un par groupe de travail, sa position exacte dans le réseau doit être connue de tous les nœuds et elle doit rester fixe tout au long du traitement de la requête. Ces contraintes fragilisent grandement le fonctionnement de la GRID dans un environnement dynamique. Si l'hôte se déconnecte, le groupe de travail sera complément paralysé le temps qu'un hôte de secours (s'il y en a) prenne le relais.

Comme deuxième révision apportée, nous proposons de répandre le service de distribution de tâches à tous les nœuds du groupe de travail. Néanmoins, le poste de nœud administrateur ne sera pas aboli. Il servira de relais entre le client qui a soumis une requête à traiter et le groupe de travail de la GRID. Il sera également le point de retour des résultats de celle-ci. De plus, la position de l'hôte n'a plus besoin d'être connu à l'avance par le client, le premier nœud de la GRID qui recevra une requête à traiter du client deviendra l'hôte du groupe de travail qu'il formera. La Figure 3.1 présente la comparaison entre les deux modèles de distribution de tâches.

Dans les deux modèles exposés, le client soumet une requête à traiter à l'hôte (H). Nous pouvons identifier clairement dans la troisième génération, la topologie des nœuds centralisés autour de l'hôte. Comparativement à cette configuration, la topologie du modèle révisé n'est plus

centralisée. Un nœud voisin de l'hôte (R) prend la responsabilité de redistribuer les sous-tâches à un sous-groupe de nœuds plus éloignés; il devient alors un relais. Le choix stratégique de ce nœud de relais ainsi que le débat sur les avantages de l'ajout de cette habilité sera abordé dans la section 3.3.3 de ce chapitre. Cette modification permet de généraliser et de contrôler la redondance des informations sur les sous-tâches à réaliser. La déconnexion de l'hôte n'est donc plus critique pour le groupe de travail.

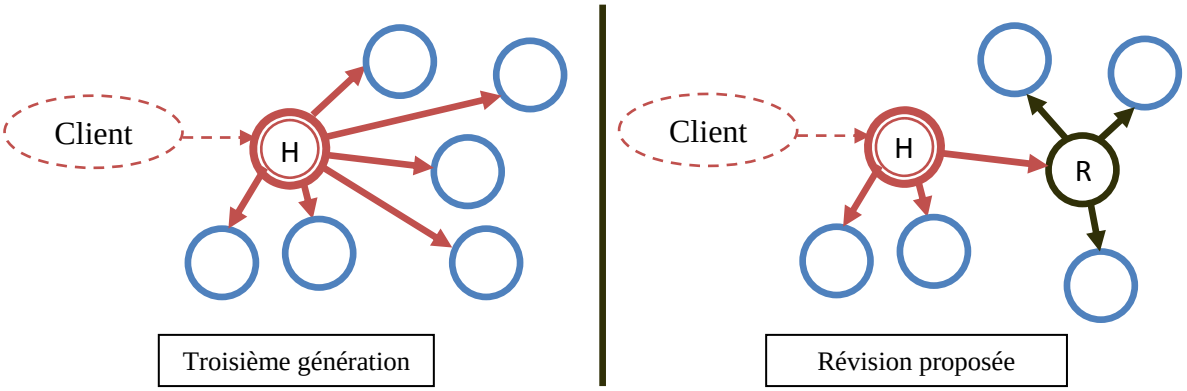


Figure 3.1 - Modèles de soumission de tâches

Nous pouvons noter que la mobilité des tâches se rapproche du concept des agents mobiles à l'exception qu'il n'y pas de négociation des requis (dans notre cas, des services demandés) entre l'agent (dans notre cas, la tâche) et la plateforme (dans notre cas, le nœud où se trouve la tâche à exécuter) à chaque fois que l'agent se déplace. Initialement toutes les plateformes (dans notre cas, les nœuds du groupe de travail) répondent aux requis de l'agent (dans notre cas, offres les services et les QdS nécessaires) pour qu'il puisse s'exécuter correctement.

Pour permettre le support de la mobilité des tâches, nous proposons d'apporter des modifications à la couche sécurité et de services.

3.2.2.1 Couche : Sécurité

Précédemment, seul un nœud était autorisé à soumettre des sous-tâches aux autres nœuds du groupe. La mobilité des tâches ne peut être possible que si les nœuds du groupe acceptent de recevoir des tâches d'un autre nœud que l'hôte. Pour cela, il faut modifier l'élément « autorisation » de cette couche.

Autorisation

Étant donné que les nouveaux nœuds pouvant soumettre des tâches ne sont pas connus à l'avance, l'autorisation de recevoir des sous-tâches d'un autre nœud sera donnée par l'hôte.

Autrement dit, si un nœud du groupe de travail a été choisi pour redistribuer des tâches ou sous-tâches, l'hôte devra prévenir tous les nœuds du groupe de travail qu'il autorise ce nœud à transférer des tâches ou à soumettre des sous-tâches tout au long de la vie du groupe de travail. Ces nouvelles informations seront stockées dans la couche de service, mais c'est cet élément qui prendra la décision d'autoriser ou pas la mobilité des tâches.

3.2.2.2 Couche : Services

Dans cette couche, le support de la mobilité des tâches entraîne une autre révision du catalogue des nœuds de l'élément d'« informations », une révision de l'entrepôt et de la cache ainsi que l'ajout d'un service de coordination.

Informations

Pour autoriser la mobilité des tâches, il est nécessaire d'ajouter une information supplémentaire dans le catalogue des nœuds sur les autorisations d'échanges entre les nœuds d'un même groupe de travail :

- Privilège des nœuds : Cette information permet de spécifier pour chaque nœud d'un groupe de travail lequel est autorisé à redistribuer et transférer des tâches ou pas.

Le catalogue des données doit aussi permettre de distinguer les différents types de tâches, entre autres les tâches à traiter localement, à transférer ou à redistribuer. Pour ce faire, nous y ajoutons aussi une information supplémentaire :

- Type de tâche : Cette information permet de spécifier pour chacune des tâches contenues dans l'entrepôt si elles sont à traiter localement, à transférer ou à redistribuer.

Entrepôt et cache

Toutes les tâches reçues n'ont plus la finalité d'être traitées localement. Il faut donc stocker les nouvelles informations relatives à la redistribution ou au transfert de la tâche. Nous devons donc ajouter un cinquième type de donnée définissant une tâche :

- Donnée relative à la mobilité : Nous y trouvons toutes les informations sur la mobilité de la tâche et sa destination finale :
  - si la tâche est à redistribuer, les directives sur la sous-division de la tâche et les contraintes d'exécution à satisfaire associées y sont sauvegardées;

- sinon, une tâche est à traiter localement, elle ne comportera aucune donnée de ce type.

Service de coordination

Ce nouveau service a deux rôles à jouer. Tout d'abord, il doit renvoyer aux bons nœuds les tâches à transférer afin qu'elles arrivent rapidement à destination. En parallèle, il traite les tâches stockées dans l'entrepôt à redistribuer. Il ne s'agit pas là de les exécuter, mais de prévoir les ressources nécessaires à leurs réalisations et finalement de collecter les résultats retournés de leurs exécutions. En bref, il effectue les opérations suivantes :

a) choisir la tâche la plus ancienne à redistribuer se trouvant dans l'entrepôt;
b) valider ses requis;
c) définir ses besoins;
d) diviser la tâche en sous-tâches;
e) envoyer des sous-tâches aux nœuds :
    - si le nœud fait déjà partie d'un groupe de travail, il envoie les sous-tâches aux nœuds libres les plus proches ou au besoin il peut en inviter;
    - sinon, il ne fait pas déjà partie d'un groupe de travail, il invite les nœuds de la GRID libres de former un groupe de travail.
f) suivre de la réalisation des sous-tâches par les nœuds précédemment contactés;
g) collecter des résultats de l'exécution dans le format demandé puis les retourner au nœud ayant soumis la tâche ou au client.

Le support de la mobilité des tâches permet donc d'envisager des mécanismes plus avancés de distribution des tâches selon l'environnement d'utilisation de la GRID. Une bonne stratégie dans le choix des nœuds autorisés à redistribuer les tâches pourra éventuellement améliorer l'efficacité de la GRID. Dans la section suivante, nous proposons les nouveaux concepts qui nous conduisent à la prochaine génération de GRID mobile.

3.3 Les nouveaux concepts

Nous avons précédemment présenté l'architecture des nœuds que nous conservons de la troisième génération de GRID. À cette architecture nous avons ajouté le support de la mobilité des nœuds et des tâches. Nous n'avons pas catégorisé ces révisions comme nouveau concept

appliqué à l'architecture des GRID car ils n'impliquaient que des modifications et la réadaptation des comportements et des mécanismes déjà existants. Toutefois, ces révisions nous ont permis de préparer l'architecture à l'ajout de vrais nouveaux concepts justifiant le changement de génération de GRID.

Dans la dernière section de ce chapitre, nous allons présenter les trois nouveaux concepts que nous nous proposons d'introduire à l'architecture d'une GRID. Le premier est une amélioration aux supports de mobilité des nœuds et des tâches. Il s'agit du transfert de l'avancement des traitements de tâches. Le deuxième est une augmentation de l'autonomie d'un nœud en introduisant le travail hors groupe et pair-à-pair. Et pour finir, la dernière proposition est un concept issu du domaine des réseaux de capteurs permettant de déterminer plus efficacement la topologie des groupes de travail selon l'environnement dans lequel ils sont hébergés.

La mise en œuvre de ces concepts permettra d'améliorer grandement le fonctionnement de notre architecture GRID dans un environnement dynamique et mobile.

3.3.1 Transfert de l'avancement des traitements de tâches

Dans un cluster, les ressources opèrent généralement dans un environnement contrôlé et stable : le nombre de ressources est fixé, leurs caractéristiques physiques telles que leurs puissances de calcul sont connues et leurs performances sont garanties. Les ressources d'un cluster sont dédiées aux tâches du cluster. Or, les ressources d'une GRID sont très souvent partagées et indépendantes. De là émerge un des plus grands obstacles à l'utilisation des GRID : pourquoi transmettre du travail à des ressources sachant qu'elles peuvent quitter à n'importe quel moment la GRID ?

La première réponse à cette question a été de faire confiance aux intentions de partage des ressources par leurs propriétaires. L'idée générale est que si nous permettons le partager d'une ressource informatique, c'est que nous savons qu'elle n'est pas souvent utilisée. La capacité des GRID à rassembler un grand nombre de ressources permet de rendre cette première réponse intuitive praticable. Plus y a des ressources, plus nous avons de chance qu'une tâche soit réalisée. Ce n'est que dans les GRID de troisième génération que nous commençons à faire face à cette réalité. La couche ressource de l'architecture des nœuds évalue l'utilisation du matériel et estime quels services pourraient être offerts selon l'historique de partage de la ressource entre la GRID et ses utilisateurs réels. Cette première approche permet localement d'estimer s'il est intéressant ou pas de partager cette ressource, limitant ainsi les déconnexions liées à l'arrêt de son partage.

Ce type de déconnexion est dit *doux* car le nœud n'est pas réellement forcé de quitter la GRID, le choix d'arrêter son partage est fait pour ne pas dégrader sa rapidité d'interaction avec son utilisateur réel. Le but de la GRID n'est pas de ralentir l'expérience avec sa ressource informatique, mais de l'exploiter pendant ses périodes d'inactivités.

Le premier concept que nous proposons d'introduire dans l'architecture de l'intergiciel des nœuds est la possibilité de transférer l'avancement de la réalisation des tâches en cours dans le nœud. Cette habilité permettra à un nœud devant effectuer une déconnexion douce, de transférer à un autre nœud du groupe de travail l'avancement de ses travaux avant de le quitter. Son application a pour objectif de minimiser encore une fois les retards engendrés par des déconnexions de nœud. Le support de la mobilité des nœuds introduit précédemment a déjà permis de réduire les retards dans la réalisation d'une tâche suite à la déconnexion temporaire du nœud auquel elle était assignée.

L'analyse de la relation (3.4) permet d'identifier deux possibilités de retard suite à une déconnexion d'un nœud :

- si le nœud se reconnecte assez rapidement au groupe de travail, plus précisément quand $T_{DelaiReconnexion} \leq T_{SousTachePerdu}$, le retard est seulement équivalent à la durée de la déconnexion du nœud $T_{DelaiReconnexion}$;
- sinon, le temps perdu est celui du délai $T_{SousTachePerdu}$ qui a déjà été alloué au traitement de la sous-tâche du nœud qui s'est déconnecté.

La mise en pratique de ce nouveau concept de transfert de l'avancement des traitements de tâches permet de minimiser l'impact de ces retards en différenciant les types de déconnexion. Dans le cas d'une déconnexion douce, le retard sera encore une fois minimisé. A l'aide du premier concept proposé, nous obtenons le temps moyen perdu $T_{C1_{RequetePerdu}}$:

$$\begin{aligned} T_{C1_{RequetePerdu}} = {} & P_{Deconnexion} \times N_{noeud} \\ & \times \left(\frac{P_{DeconnexionDouce} \times T_{TransfertAvancement}}{N_{noeud}} \right. \\ & \left. + (1 - P_{DeconnexionDouce}) \times T_{C1_{DeconnexionForte}} \right) \end{aligned} \tag{3.5}$$

Où :

- $T_{C1_{DeconnexionForte}}$ représente le temps moyen perdu engendré par une déconnexion forte :

$$T_{C1_{DeconnexionForte}} = \left(\frac{T_{SousTachePerdu} + T_{Redistribution}}{N_{noeud}}\right) + P_{Reconnexion} \times \left(\frac{T_{DelaiReconnexion} - T_{SousTachePerdu} - T_{Redistribution}}{N_{noeud}}\right) \quad (3.6)$$

- $P_{DeconnexionDouce}$ représente la probabilité que la déconnexion soit douce;
- $T_{TransfertAvancement}$ représente le délai de temps nécessaire pour transférer l'état de l'avancement de la tâche.

Cette reformulation du temps moyen perdu permet d'identifier une troisième possibilité de retard dont la durée peut être négligeable $T_{TransfertAvancement}$. En effet, le transfert de l'avancement du traitement d'une tâche consiste à envoyer la réponse incomplète ainsi que les valeurs des variables utilisées. La quantité de ces données est généralement nettement moins volumineuse que la tâche elle-même (soit les fichiers requis, les codes informatiques, les librairies, les directives sur la réalisation de la tâche…). Nous pouvons donc poser la relation (3.7) :

$$0 < T_{TransfertAvancement} \leq T_{Redistribution} \ll T_{SousTache} \quad (3.7)$$

Ce troisième scénario permet de minimiser encore une fois le temps moyen de retard dû à une déconnexion. En le réduisant à un délai peu significatif. La mise en pratique de cette proposition nécessite des modifications à la couche des ressources et de services

3.3.1.1 Couche : Ressources

C'est aux gestionnaires des ressources physiques du nœud de prendre la décision de retirer celui-ci du groupe de travail avant de dégrader la qualité de l'expérience d'utilisation de son propriétaire et de déclencher le transfert des tâches à d'autres nœuds. Pour ce faire, les mesures de l'unité de calcul ne suffisent plus, il faut analyser plus en détail les processus s'exécutant sur la machine. Nous ajoutons donc un nouvel élément à cette couche

Gestionnaire de processus

Le rôle de cet élément est de recenser les différents processus en cours ainsi que leurs priorités d'exécution. De cette manière, il sera possible de détecter automatiquement l'arrivée d'un utilisateur devant la machine ou à distance. Selon ses droits, le partage de la ressource pourrait cohabiter avec son utilisation ou être forcé de s'arrêter.

À l'aide du gestionnaire de l'unité de calcul, si la charge de travail des CPU est à leur maximum et que l'exécution de processus prioritaires est retardée, cet élément émettra un signal à la couche de service pour arrêter l'exécution des tâches reçues par la GRID.

3.3.1.2 Couche : Services

Le signal d'initiation d'une déconnexion douce du nœud étant émis par la couche des ressources, c'est la couche des services qui est responsable du transfert de l'avancement de la tâche. Le partage des ressources peut se faire de différentes manières : stockage de données, base de données, serveur Web, agent et tous autres services possibles. Dans ce mémoire, nous ne considérons que le partage de la puissance de calcul et le transfère de l'avancement de ce type de tâche.

Gestionnaire de tâches

Précédemment, nous avons indiqué que cet élément avait pour rôle d'exécuter les tâches reçues de la GRID, c'est donc cet élément qui consomme principalement la puissance de calcul de la ressource. Dans le cas où il faut céder les unités de calcul de la ressource à des processus plus prioritaires, son utilisation de la puissance de calcul doit cesser rapidement. Le signal d'initialisation de déconnexion douce avec le groupe de travail est traité de la même manière qu'une déconnexion réelle au réseau : le traitement des tâches en cours est mis en veille. À la différence qu'elles sont stockées dans l'entrepôt comme des tâches prioritaires à transférer au nœud les ayant soumis.

Service de coordination

Ce service ayant aussi reçu le signal de déconnexion douce du gestionnaire de tâches, il s'assure que toutes les tâches prioritaires à transférer sont émises aux bons nœuds, réalisant ainsi le transfert de l'avancement des tâches en cours.

Tant que ce service n'a pas reçu un signal d'arrêt qui lui est destiné, il poursuivra son fonctionnement normal même si le gestionnaire de tâche est arrêté; cela signifiera que le nœud ne traitera aucune tâche et ne servira que de relais. Si la consommation des ressources physiques est toujours trop élevée, la couche des ressources pourrait aussi demander son arrêt complet. Dans ce cas, toutes nouvelles tâches à redistribuer ou à transférer seront ignorées. Les tâches restantes à transférer seront envoyées et tous les nœuds ayant soumis des tâches à redistribuer seront avertis de l'arrêt de ce service ainsi que de la liste des nœuds que coordonnait ce nœud.

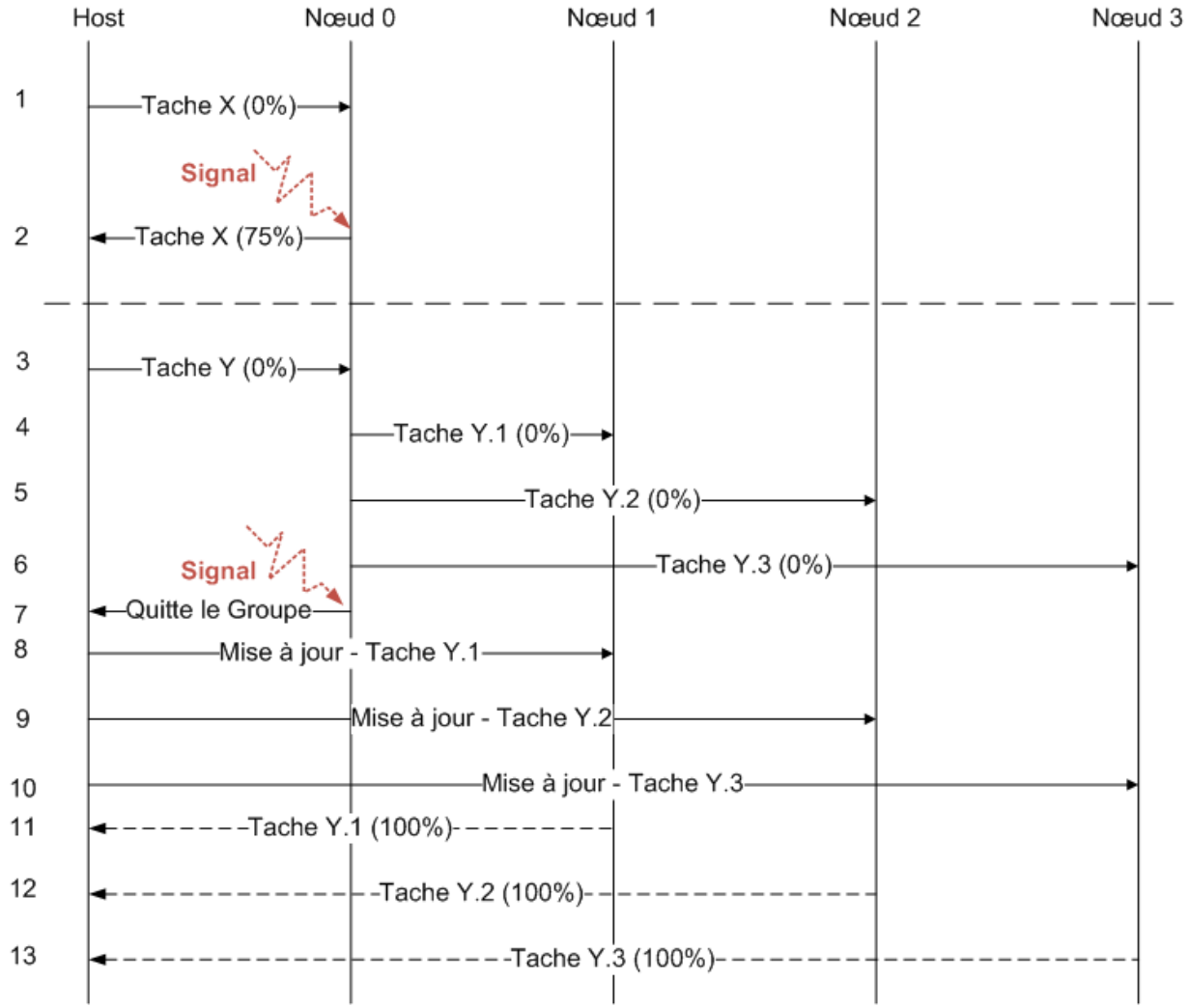


Figure 3.2 - Diagramme de séquence d'une déconnexion douce

Pour schématiser une déconnexion douce, nous présentons à la Figure 3.2, le diagramme séquentiel de deux scénarios. Dans le premier scénario, l'hôte envoie une tâche X au nœud 0 à réaliser dans la transaction (1). Une fois que le signal de déconnexion douce est émis par le gestionnaire de processus, dans la transaction (2), le gestionnaire de tâche met en veille

l'avancement à 75 % du traitement de la tâche qui sera envoyé à l'hôte par le service de coordination. Finalement la déconnexion aura lieu.

Dans le deuxième scénario, l'hôte envoie une tâche Y à redistribuer dans la transaction (3). Le nœud 0 la divise en trois sous-tâches et la soumet à trois autres nœuds dans les transactions (4), (5) et (6). Une fois le signal de déconnexion douce émis par le gestionnaire de processus avec l'arrêt complet du service de coordination; le nœud 0 avertit l'hôte de sa déconnexion imminente et l'informe des nœuds dont il était responsable dans la transaction (7). L'hôte étant averti de la déconnexion du nœud 0, il envoie un message de mise à jour aux trois autres nœuds dans les transactions (8), (9) et (10). Une fois les sous-tâches réalisées, dans les transactions de (11) à (13), les nœuds 1, 2 et 3 ayant reçu des messages de mises à jour renvoient leurs résultats à l'hôte au lieu du nœud 0.

Nous supposons que l'application de ce concept permet de réellement faire face à la réalité du partage temporaire des ressources. Cela en prenant en compte non seulement leurs disponibilités, mais en effectuant aussi des relevés à la fin de leurs périodes de disponibilité. Les travaux en cours des nœuds avant leurs déconnexions n'étant plus perdus, le retard engendré par leurs déconnexions douces devient donc négligeable. Cependant, dans le cas d'une déconnexion forcée, d'autres mesures sont à prendre. Ce qui nous amène au concept proposé suivant.

3.3.2 Travail "hors groupe" et "pair-à-pair"

Quelle que soit la génération précédente de GRID, la topologie du réseau durant le traitement d'une tâche s'efforce d'être centralisée. Tous les nœuds communiquent directement et seulement avec l'hôte. La centralisation du groupe de travail permet de facilement administrer la coopération entre les nœuds. L'hôte jouant le rôle d'intermédiaire, il rend transparentes les communications entre nœuds. Facilitant ainsi toutes interconnexions ainsi que la réaffectation de tâche en cas de déconnexion. Cette configuration a démontré son efficacité dans un environnement peu dynamique. La stabilité de sa topologie lui permet de s'axer sur la performance. Cette topologie est cependant inadaptée aux deux types d'environnements suivants :

- environnements dans lesquels des déconnexions sont fréquentes;
- environnements où les coûts de communications ne sont pas négligeables.

Le premier type d'environnement énuméré fait allusion à la faible autonomie des nœuds. En effet, se déconnecter de la GRID signifie une remise à zéro du nœud. Cela explique la dégradation de performance de celle-ci dans un environnement dynamique. Une première amélioration a été d'ajouter le support de la mobilité des nœuds dans notre révision des éléments de l'architecture de troisième génération. Ces modifications nous ont permis d'améliorer le comportement du nœud suite à une déconnexion en mettant en veille les traitements en cours au lieu de les arrêter.

En second lieu, la configuration figée et centralisée de la topologie est inappropriée à un environnement en considérant les coûts de communication entre les éléments de la GRID. Les connexions aux hôtes devant être continues, les interconnexions entre des nœuds distants et l'hôte pourraient être coûteuses dans certains réseaux de communication. Les opérateurs et fournisseurs d'accès mobile limitent encore très souvent la bande passante et la quantité de données pouvant être échangées sans engendrer de frais supplémentaires aux tarifs déjà fixés. Les nœuds se trouvant dans un environnement mobile pourraient donc entraîner des frais d'exploitation de la GRID non négligeables. Il serait donc plus avantageux dans des environnements dynamiques et mobiles de ne plus s'efforcer de garder une topologie centralisée. Quand le traitement de tâche ne nécessite pas de coopération, les nœuds n'ont pas un besoin réel de rester connectés à l'hôte. Dans le cas contraire, ils pourraient interagir directement avec un nœud proche au lieu de faire tout le tour des réseaux, limitant ainsi les frais liés à l'itinérance.

Comme deuxième concept, nous voulons améliorer l'autonomie des nœuds lorsqu'ils sont déconnectés de leurs groupes de travail. Nous proposons donc de permettre aux nœuds de poursuivre leurs réalisations en cours lorsqu'ils sont hors du groupe de travail, mais aussi de considérer les coopérations pair-à-pair. Les nœuds ne seront donc plus obligés de rester connectés autour de l'hôte. Une fois leurs tâches reçues, ils pourront se déconnecter du groupe du travail, commencer à les traiter, coopérer avec leurs voisins puis transmettre leurs résultats à l'hôte.

L'habilité d'un nœud à pouvoir travailler hors de son groupe permet dans certains cas de ne plus ralentir le traitement d'une requête suite à une déconnexion forcée. Elle est une amélioration du support de la mobilité des nœuds introduit précédemment. Nous pouvons redéfinir à partir de la relation (3.5), le temps moyen perdu durant le traitement d'une requête $T_{C2_{RequetePerdu}}$.

$$T_{C2_{RequetePerdu}} = P_{Deconnexion} \times N_{noeud} \times \left(\frac{P_{DeconnexionDouce} \times T_{TransfertAvancement}}{N_{noeud}} + (1 - P_{DeconnexionDouce}) \times T_{C2_{DeconnexionForte}} \right) \quad (3.8)$$

Où :

- $T_{C2_{DeconnexionForte}}$ représente le temps moyen perdu engendré par une déconnexion forte avec l'application du deuxième concept :

$$T_{C2_{DeconnexionForte}} = \left(\frac{T_{SousTachePerdu} + T_{Redistribution}}{N_{noeud}} \right) (1 - P_{Reconnexion}) \quad (3.9)$$

Suite à l'application du deuxième concept, dans l'estimation du temps moyen perdu, nous pouvons noter la disparition du terme $T_{DelaiReconnexion}$de la relation (3.6). Précédemment, un nœud est réadmis au groupe de travail si $T_{DelaiReconnexion} < T_{SousTachePerdu}$ et engendre un retard de $T_{DelaiReconnexion}$. Dans $T_{C2RequetePerdu}$, un nœud ayant subi une déconnexion forcée est réadmis s'il se reconnecte avant $T_{SousTache}$ et cela, sans ajouter de délai supplémentaire. De plus, la période de temps $T_{SousTache}$ étant toujours plus longue que $T_{SousTachePerdu}$, la probabilité de reconnexion d'un nœud $P_{Reconnexion}$ est augmentée.

L'habilité des nœuds à pouvoir coopérer de pair-à-pair permet de réduire le délai de communication entre deux nœuds. L'hôte n'étant plus le relais, il n'est plus un goulot d'étranglement ni un point critique au fonctionnement du groupe de travail. Il n'est plus nécessaire d'entretenir des liens avec tous les nœuds du groupe de travail pendant le traitement d'une requête d'un client. Les communications sont maintenant directes entre les nœuds et se créent aux besoins. Cette topologie est donc plus avantageuse dans un environnement dynamique et mobile, car il permet de granuler la topologie du groupe de travail. La Figure 3.3 présente la comparaison entre une topologie centralisée et décentralisée.

Dans les deux modèles exposés, le client soumet une requête à traiter à l'hôte (H). Le traitement de la requête commence après que toutes les tâches soient distribuées aux nœuds du groupe de travail (voir la Figure 3.2). Dans la modélisation d'une topologie centralisée, l'hôte a le

contrôle continu de tous les nœuds du groupe de travail durant toute la durée du traitement. Si le nœud (1) a besoin de coopérer avec le nœud (2), il doit passer par l'hôte même s'il est éloigné d'eux. Dans cet exemple, l'interconnexion entre le nœud (1) et l'hôte (H) entraîne un délai de réponse de 123 ms et celui de l'hôte (H) au nœud (2) entraine un de 97 ms. Nous obtenons un temps de réponse de 220 ms à chaque échange d'information entre les nœuds (1) et (2) et cela même s'ils sont dans le même réseau. Dans une topologie décentralisée, le nœud (1) communique directement avec le nœud (2). Cela réduit ainsi le temps de réponse à 20 ms. Dans ce modèle, les interconnexions sont créées au besoin et maintenues temporairement. La topologie est donc en constante évolution au long du traitement.

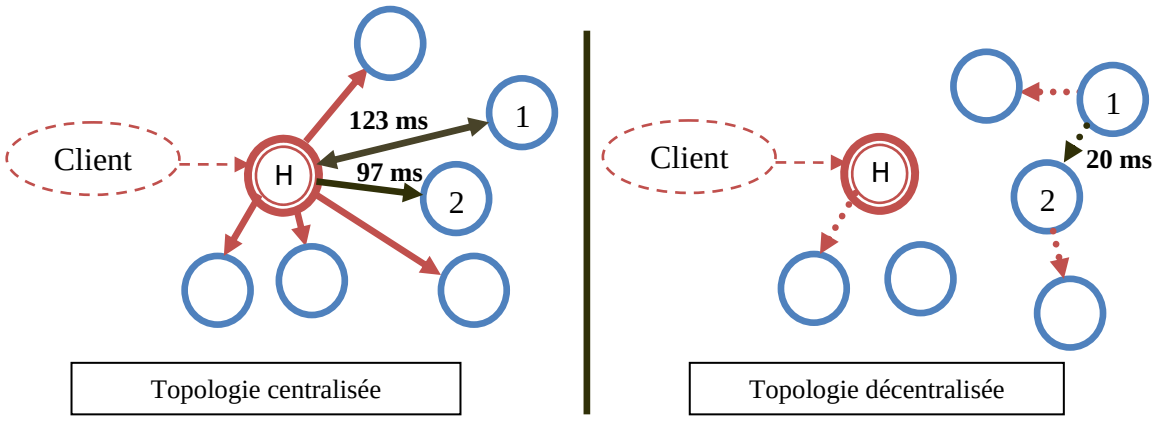


Figure 3.3 - Topologie des nœuds durant un traitement de requête

Pour appliquer ce deuxième concept, des éléments devront être ajoutés à la couche du réseau et de services de l'architecture de l'intergiciel des nœuds. Certains des éléments déjà présents devront être aussi adaptés pour s'assurer de leurs compatibilités avec les nouvelles habilités de travail hors groupe et pair-à-pair.

3.3.2.1 Couche : Réseau

La couche réseau compte déjà trois éléments. Leurs rôles sont de permettre la découverte de nouveaux nœuds, de gérer les interconnexions dans un groupe de travail et de transmettre les données aux bons destinataires. Dans notre architecture proposée, la couche du réseau a un nouveau rôle à jouer : elle doit évaluer la topologie et les performances des réseaux contenant les nœuds.

Découverte de la topologie et mesure de la qualité de service

Ce nouvel élément est très particulier, car il ne s'intéresse qu'aux réseaux hébergeant les nœuds du groupe de travail. Il ne participe pas directement au traitement de requête. Son rôle est d'évaluer, de mesurer et de déterminer la position et les mouvements des autres nœuds et de l'hôte. Il permet d'évaluer la performance des différents chemins de communication avec les autres nœuds. Périodiquement, il échange ces informations avec ses nœuds voisins de la GRID, c'est-à-dire les nœuds qui lui sont directement accessibles.

3.3.2.2 Couche : Services

La couche des services représente toute la logistique de l'intergiciel d'une ressource informatique de la GRID. L'habilité de travailler hors du groupe y est ajouté par modification du gestionnaire de tâche. De plus, cette couche doit conserver dans son élément d'« informations », les nouvelles mesures de la qualité de service des liens de la topologie de la GRID effectuée par la couche du réseau. Finalement dans une topologie décentralisée, le besoin de coopération devra être géré localement. Pour ce faire, un nouvel élément nommé P2P sera introduit.

Gestionnaire de tâches

Les modifications précédentes ont permis aux nœuds d'un groupe de travail de se déconnecter temporairement du réseau, de changer de réseau et même d'adresse physique. Le support de la mobilité d'un nœud était l'initiation au travail hors groupe. L'hôte n'étant plus responsable de la synchronisation et de la coopération entre nœuds, il n'est plus nécessaire de mettre en veille la tâche en cours suite à une déconnexion.

Durant le travail hors groupe, le besoin d'une tâche de coopérer avec les tâches d'autres nœuds sera transmis à l'élément P2P. C'est cet élément qui se chargera d'établir les liens avec les nœuds visés et de les rendre transparents.

Informations

Les mesures de performance des liens doivent être associées aux informations contenues dans le catalogue des nœuds. Pour cela, nous devons ajouter de nouveaux champs d'information à chaque nœud du groupe de travail et de la GRID :

- Temps de réponse du lien direct : Indice de performance mesuré en ms représentant le délai entre l'envoi et la réception d'un court message à ce nœud.

- Débit du lien : Indice de performance mesuré en Mo/s représentant la quantité de données en mégaoctet pouvant être transmise en une seconde.
- Historique des déconnexions : Moyenne du délai de temps pendant lequel le nœud reste connecté au groupe de travail sans interruption.
- Table de routage : Chemins des nœuds par lesquels il faut passer pour envoyer un message à autre un nœud de la GRID. Il est évalué à l'aide de la méthode « des vecteurs de distance ».

P2P

Ce nouvel élément représente le centre de coordination des interconnexions de pair-à-pair. En se basant sur les indices de performance des réseaux mesurés par sa couche réseau et celle qu'il a reçue, il établit des liens avec d'autres nœuds du groupe de travail. Selon les requis nécessaires à la réalisation d'une tâche, cet élément peut être amené à définir des tables de routage avancées minimisant les temps de réponses de la topologie des nœuds. Cependant, à ce stade de la présentation de notre nouvelle architecture GRID, son rôle se limitera à déterminer puis à initialiser la création de liens directs avec les autres nœuds.

Jusqu'à présent, l'architecture des GRID ne prenait pas en compte les caractéristiques des réseaux dans lesquelles elle fonctionnait. Ce deuxième concept nous a orientés vers un modèle de groupe de travail pouvant être décentralisé et à l'écoute de son environnement. Même si nous pouvons calculer les performances à un instant donné de la topologie du groupe de travail, dans un environnement dynamique et mobile ils seront en constante évolution. Pour minimiser les efforts de mise à jour et de synchronisation, il faudrait être capable de prédire leurs évolutions. L'organisation de la topologie du groupe de travail devient alors plus qu'intuitive.

3.3.3 Topologie du groupe de travail adaptée aux environnements

Les recherches menées depuis les premières générations de GRID ont mis en évidence deux critères d'amélioration à apporter, la première étant de généraliser la structure même des tâches traitées par les GRID. Cela dans l'optique d'ouvrir les GRID à un grand nombre d'applications. La deuxième quant à elle, vise à décentraliser la topologie des GRID afin d'augmenter le nombre de ressources pouvant s'y rejoindre. C'est à partir des GRID de troisième génération que ces améliorations ont porté leurs fruits. Les tâches sont divisées en sous-tâches et les GRID en groupes de travail. La capacité de cette génération à pouvoir s'étendre au-delà des limites

traditionnelles engendre une nouvelle problématique non plus fonctionnelle, mais cybernétique. C'est-à-dire, que nous ne nous intéressons plus aux fonctionnements propres des éléments de la GRID, mais à leurs interactions menant à la réalisation globale de la requête soumise à l'hôte. D'où notre intention de prendre en considération l'environnement et le type de tâches à réaliser afin de déterminer la topologie la mieux adaptée à chaque groupe de travail pour réaliser efficacement les requêtes soumises.

L'architecture des nœuds de troisième génération héritée, améliorée puis complétée par les deux concepts précédents donne aux nœuds une plus grande autonomie. Ils sont désormais capables de réaliser les opérations suivantes au niveau du réseau :

- ✓ découvrir d'autres nœuds de la GRID;
- ✓ former un groupe de travail;
- ✓ rejoindre ou se déconnecter d'un groupe de travail;
- ✓ changer de réseau qui l'héberge.

Ainsi que les opérations fonctionnelles suivantes :

- ✓ transférer une tâche;
- ✓ diviser une tâche en sous-tâches;
- ✓ transférer l'avancement d'une tâche;
- ✓ réaliser une tâche en groupe ou individuellement (hors groupe);
- ✓ réaliser une tâche en collaborant directement avec d'autres nœuds (pair-à-pair).

Ces comportements ont permis au groupe de travail de se remettre plus rapidement des déconnexions des nœuds le constituant et même dans certains cas de les ignorer. Malgré cela, les contraintes de connectivités et de mobilité peuvent sérieusement réduire la performance des GRID dans des environnements dynamiques et mobiles. Un scénario « catastrophe » serait un groupe de travail dans lequel l'hôte est en mouvement continu. Chaque déconnexion de l'hôte, signifierait que tous les nœuds subissent une déconnexion forcée. Le choix de l'hôte étant prédéfini par le client soumettant la requête, il ne prend pas en compte sa position actuelle dans l'environnement où il se situe. Ce scénario met donc en évidence l'importance de prendre en considération l'environnement dans lequel le groupe de travail se situe afin de s'y adapter efficacement. Ce troisième concept va donc au-delà de la décentralisation des groupes de travail et propose d'organiser leurs topologies selon les besoins des tâches et les contraintes de l'environnement.

3.3.3.1 Les indices d’évaluation

Dans le domaine des réseaux de capteurs, les contraintes d’autonomies et de portées ont poussé les chercheurs à proposer des méthodes pour former des chemins de communications efficaces et économiques en énergie. Fabien Nimbona et S. Pierre [24] proposent de former des petits groupes de capteurs appelés « grappe » et d’y définir un responsable par regroupement nommé « tête de grappe ». La hiérarchie d’une tête de grappe est évaluée à l’aide d’un indice d’élection représentant la performance du capteur ainsi que la fiabilité de ces interconnexions.

Comme troisième et dernier concept, nous nous proposons d’étendre la notion de grappe au groupe de travail. Nous supposons qu’en divisant un groupe de travail dans un environnement globalement dynamique et mobile en plusieurs petites grappes plus stables, nous réduirons les impacts des déconnexions et nous optimiserons les communications entre nœuds d’une même grappe. Cette décentralisation permettra tout d'abord d'alléger l’hôte puisqu’il ne communiquera qu’avec les têtes de grappes. Les têtes de grappe seront maintenant responsables de distribuer et collecter les informations de leurs grappes, réduisant ainsi les dégradations de performance liées au mouvement de l’hôte. De plus, La redondance des tâches à effectuer permettra d’accélérer la redistribution des tâches en cas de déconnexion ou la synchronisation suite à une reconnexion.

Les nœuds disposant déjà de toutes les fonctionnalités pour appliquer ce concept, nous définissons les indices d’évaluation qui nous permettrons de les évaluer de manière efficace. Pour ce faire, nous allons définir les indices suivants.

Indice de performance

Vecteur représentant la puissance de calcul du nœud, la taille de sa mémoire virtuelle disponible, l’espace disque disponible ainsi que son autonomie énergétique (énergie résiduelle).

Indice de mobilité

Probabilité d’une déconnexion du nœud avec son groupe de travail durant le traitement d’une requête.

Indice de la qualité de service de la connexion

Vecteur représentant le délai de réponse et le débit d’une connexion entre deux nœuds. Il y a un indice par sens de communication, soit deux par lien.

Indice d'élection

Valeur représentant le potentiel d'un nœud de la GRID à être élu comme tête de grappe. Contrairement aux indices précédents, il n'est pas évalué et retransmit périodiquement. Pour pouvoir déterminer cet indice, il est nécessaire de connaître le type d'application qui sera traitée par le groupe de travail, nous pouvons donc l'évaluer qu'au moment d'élire des têtes de grappe. Pour faciliter sa représentation, nous posons les fonctions $P(n)$, $M(n)$ et $Q(n,n')$.

La fonction $P(n)$ permet d'évaluer la performance d'un nœud à l'aide de son vecteur d'indice de performance. Elle est représentée par la relation (3.10) :

$$P(n) = I_{Performance_{CPU_n}} \times \alpha_1 + I_{Performance_{RAM_n}} \times \alpha_2 + I_{Performance_{HD_n}} \times \alpha_3 + I_{Performance_{E_n}} \times \alpha_4 \quad (3.10)$$

Où :

- $I_{Performance_{CPU_n}}$ représente la puissance de calcul en GHz du nœud *n*;
- $I_{Performance_{RAM_n}}$ représente la mémoire virtuelle disponible en Go du nœud *n*;
- $I_{Performance_{HD_n}}$ représente l'espace disque disponible en Go du nœud *n*;
- $I_{Performance_{E_n}}$ représente l'autonomie restant en pourcentage du nœud *n*;
- α_1, α_2, α_3 et α_4 représente les coefficients assignés à chaque élément de l'indice de performance d'un nœud. Leurs valeurs varient selon le type d'application à traiter. Par exemple pour une application de calcul instance $\alpha_1 > \alpha_2 > \alpha_4 > \alpha_3$. Ou encore pour une application de stockage $\alpha_3 > \alpha_2 > \alpha_4 > \alpha_1$;

La fonction $M(n)$ permet d'évaluer l'indice de mobilité d'un nœud *n* à l'aide de l'historique des déconnexions :

$$M(n) = I_{Mobilite_n} = \frac{\frac{N_{operations}}{\sum_{x=1}^{v} P(n_x)} \times 100}{T_{Avant\_deconnexion_n}} \quad (3.11)$$

Avec :

- $N_{operations}$ représentant le nombre total d'instructions informatiques nécessaires pour traiter une requête. Ce nombre est estimé à partir de la complexité de la requête à traiter et de son code source;

- $\sum_{x=1}^{v} P(n_x)$ représentant la somme des indices de performances des *v* nœuds $n_1, n_2, n_3, n_4 \ldots n_x$ du groupe de travail;
- $T_{Avant\_deconnexion\,n}$ représentant le délai de temps moyen avant une déconnexion du nœud *n*. Il est estimé à l'aide de l'historique des périodes de temps pendant laquelle le nœud reste connecté à la GRID sans interruption.

La fonction $Q(n, n')$ permet d'évaluer la qualité de service d'une connexion entre deux nœuds voisins *n* et *n'* de la GRID. Et cela à l'aide du vecteur de l'indice de la qualité de service du lien $I_{QdS\,nn'}$:

$$Q(n, n') = I_{QdS_{Debit\,nn'}} \times \beta_1 + I_{QdS_{Delai\,nn'}} \times \beta_2 \tag{3.12}$$

Où :

- $I_{QdS_{Debit\,nn'}}$ représente le débit du lien du nœud *n* à *n'*.
- $I_{QdS_{Delai_{nn'}}}$ représente le délai de réponse du lien du nœud *n* à *n'*.
- $\beta_1$et β_2 représente les coefficients assignés à chaque élément de l'indice de la qualité de service du lien.

Nous pouvons maintenant exprimer l'indice d'élection d'un nœud *n* demandé par une tête de grappe *t* à l'aide la de la fonction *E(n)* :

$$E(n) = I_{Election_n} = \frac{\left(\sum_{x=1}^{v} \frac{P(n_x)}{Q(n, n_x)^{C_Q} \times M(n_x)^{C_M}} + P(n)\right)^{C_P}}{Q(n, t)^{C_Q} \times M(n)^{C_M}} \tag{3.13}$$

Avec :

- n_x représentant le Xème nœud des *v* voisins du nœud *n* excluant le nœud *t*.
- C_P, C_M et C_Q représentant les coefficients attribués respectivement à l'importance de la performance, de la mobilité et de la qualité de service des connexions des nœuds. Ces coefficients varient selon le type d'application. Par exemple, pour traiter une requête divisible en un ensemble de tâches indépendantes, Nous aurons $C_P > C_Q > C_M$ .Ou encore pour une requête nécessitant une coopération entre tâches $C_M > C_Q > C_P$.

Indice de la qualité de service de la requête

Valeur représentant la performance nécessaire à atteindre par un groupe de travail pour pouvoir garantir la qualité de service recommandée pour le traitement de la requête. La performance d'un groupe de travail étant la somme de la puissance de calcul des nœuds qui le compose, cette valeur se mesure en nombre d'instructions informatiques devant être effectuées en une seconde par tout le groupe de travail.

3.3.3.2 Les stratégies de formation d'un groupe de travail

Les indices d'évaluation étant maintenant définis, nous pouvons présenter la stratégie que nous proposons pour former et organiser un groupe de travail. Les têtes de grappes élues sont les nœuds les mieux équilibrés entre leurs nombres de nœuds voisins avec qui ils peuvent communiquer, leurs performances et leurs mobilités. Pour former un groupe de travail dans l'architecture GRID que nous proposons, nous devons rajouter trois critères de sélection dans le mécanisme d'élection proposé par cet auteur :

- la qualité de service des interconnexions;
- le type de tâche à réaliser : tâche indépendante ou tâche coopérative;
- l'économisassions des ressources.

Usuellement, la formation d'un groupe de travail par l'hôte se fait simplement par le regroupement de ces nœuds voisins suite à une invitation. Chaque nœud ayant accepté de se joindre au groupe de travail entretiendra un lien direct avec l'hôte afin de former une topologie centralisée. À l'aide des indices de performance, de mobilité, de qualité de service des connexions, de qualité de service de la requête et d'élection définis précédemment, nous présentons les différentes étapes du nouveau mécanisme de formation de groupe de travail.

Étape 1

Chaque nœud de la GRID détermine son indice de performance et la qualité de service de ses connexions puis il les transmet à ses voisins lorsqu'il signale sa présence dans la GRID. S'il se déplace durant son séjour dans la GRID, le nœud met à jour les indices de qualité de services des nouvelles connexions et les partages avec ses nouveaux nœuds voisins. De même, lorsque le nœud a un nouveau voisin et qu'il reçoit ces indices de performance et de la qualité de service de la connexion, il détermine à son tour la qualité de service de la nouvelle connexion. Il l'envoie alors en plus de son indice de performance, un message en réponse à son nouveau voisin

Cette série d'échange d'indices permettra à tous les nœuds de la GRID de connaître leurs voisins et d'être prêt à la formation d'un groupe de travail.

Étape 2

À la requête d'un client, le nœud ayant reçu en premier celle-ci devient l'hôte. Il joue le rôle de la tête de grappe principale et initialise la formation de son groupe de travail.

Étape 3

L'hôte émet des invitations à tous ses nœuds voisins dans la GRID. Il les invite à rejoindre sa grappe principale qui formera son groupe de travail. Dans son message d'invitation, il indique : le type d'application qui sera à traiter, la quantité estimée d'instructions informatiques à devoir traiter par ce nœud et l'indice de la qualité de service de la requête à combler.

Étape 4

Un nœud de la GRID ayant reçu une invitation d'une tête de grappe effectue les opérations suivantes :

- Si *le nœud est déjà une tête de grappe dans ce groupe de travail* :
  - Cette tête de grappe émet elle aussi à ses voisins des invitations pour joindre sa propre grappe.
  - Pour chaque voisin, nous répétons donc la 4e étape.
- Si *le nœud est disponible ou fait déjà partie d'une grappe de ce groupe de travail* :
  - Il détermine son indice d'élection à l'aide des informations reçues sur la tâche à traiter :
    - Dans le calcul de cet indice d'élection, il ne prend en considération que les nœuds voisins disponibles, c'est-à-dire les nœuds voisins n'étant ni tête de grappe ni élément d'une grappe.
    - Si *le nœud est un élément d'une grappe*, il ne prendra pas en compte son indice de performance dans le calcul de l'indice d'élection puisqu'il fait déjà partie du groupe de travail.
  - Une fois ces indices évalués, il les retransmet à la tête de grappe qui lui a envoyé l'invitation.

Étape 5

Les têtes de grappe récoltent les réponses de leurs voisins et effectuent les opérations suivantes:

- Si *la tête de grappe est l'hôte* :
  - Elle choisit un nœud :
    - Si *la tête de grappe offre aussi le service du partage de sa puissance de calcul* :
      - Elle choisit entre elle-même ou un nœud de son voisinage, celui qui possède un indice d'élection dont la valeur est légèrement plus élevée que l'indice de qualité de service de la requête restant à combler.
      - Si *aucun ne peut être choisi*, c'est le nœud avec la valeur d'indice d'élection le plus élevé qui sera retenu.
    - Sinon, Elle choisit le nœud parmi son voisinage qui possède un indice d'élection dont la valeur est légèrement plus élevée que l'indice de qualité de service de la requête restant à combler. Si aucun ne peut être choisi, c'est le nœud avec la valeur d'indice d'élection le plus élevé qui sera retenu.
  - Si *le nœud choisi est l'hôte*, il forme sa grappe en regroupant ses voisins qui lui ont répondu.
  - Si *le nœud choisi est l'un de ses nœuds voisins*, il est élu comme nouvelle tête de grappe et dépend de celle l'ayant invité. Il forme sa grappe en y regroupant les nœuds voisins qui lui ont permis de déterminer son indice d'élection.
  - Si *le nœud choisi a été suggéré par une tête de grappe voisine*, il est élu comme nouvelle tête de grappe et il sera sous la responsabilité de la tête de grappe la plus proche de lui. Puis, la nouvelle tête de grappe forme sa grappe en y regroupant les nœuds voisins qui lui ont permis de déterminer son indice d'élection.
  - S'il n'y a *aucun choix de nœud possible*, le groupe de travail ne peut plus s'étendre sur d'autres nœuds de la GRID car il n'y en a plus de disponibles. Nous passons à la 7e étape.

Si *la tête de grappe a elle-même reçu une invitation* :

- Elle choisit un nœud parmi toutes les réponses reçues :
  - Elle choisit le nœud ayant un indice d'élections dont la valeur est légèrement plus élevée que l'indice de qualité de service de la requête restant à combler.
  - Si *l'indice de qualité de service de la requête restant ne peut pas être comblé*, elle choisit le nœud avec l'indice d'élection la plus élevée.
- Si *un nœud est choisi* :
  - Si *le nœud choisi par cette la tête de grappe t n'est pas un nœud voisin, mais plutôt un nœud suggéré par une tête de grappe voisine n* : la tête de grappe *t* réévalue l'indice d'élection du nœud choisi pour prendre en considération la mobilité et la qualité de service du lien, de la tête de grappe voisine *n* qui l'a répondu. Pour ce faire nous utilisons la relation (3.14) :

$$I_{Election\_Réévalué} = \frac{\left(I_{Election\_Recu}\right)^{C_P}}{Q(n,t)^{C_Q} \times M(n)^{C_M}} \qquad (3.14)$$

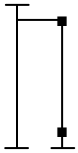
Avec :

- *n*, la tête de grappe voisine ayant suggéré un nœud à sa tête de grappe *t* hiérarchiquement plus élevée;
- $I_{Election\_Reevalue}$ représentant l'indice d'élection réévalué. Cette valeur remplacera l'indice d'élection reçu du nœud suggéré;
- $I_{Election\_Recu}$ représentant l'indice d'élection reçu du nœud suggéré par la tête de grappe voisine.

  - Elle retransmet l'indice d'élection du nœud voisin choisi à la tête de grappe qui l'a invitée.
- S'il n'y a *aucun choix de nœud possible*, le groupe de travail ne peut plus s'étendre sur d'autres nœuds de la GRID car il n'y en a plus de disponibles. Nous passons à la 7e étape.

Étape 6

La tête de grappe détermine si le groupe de travail est suffisamment performant pour réaliser la requête ou doit s'étendre encore à d'autres nœuds de la GRID. Rappelons que la performance d'un groupe de travail est la somme de la puissance de calcul des nœuds qui le compose.



Si *la performance du groupe de travail est plus élevée que l'indice de la qualité de service de la requête à garantir*, nous passons à l'étape suivante.

Sinon, nous répétons la 3e étape.

Étape 7

Finalement, la topologie actuelle du groupe de travail est suffisante ou est la meilleure possible. Nous pouvons commencer le traitement de la requête.

Sur le plan théorique, cette méthode permet de regrouper un groupe de travail en au moins une grappe, avec à sa tête le nœud le plus apte à jouer le rôle d'un hôte. Si la GRID s'étend sur plusieurs réseaux, les têtes de grappe réorganiseront les interconnexions des nœuds de manière à limiter les échanges inter-réseaux, favorisant ainsi les communications locales. Nous pouvons noter que le choix des têtes de grappe se fait suivant une heuristique gloutonne. À chaque choix à faire, nous choisissons la tête de grappe ayant la valeur d'indice d'élection la plus élevé et cela, sans revenir sur nos choix.

Mise à part l'augmentation de la flexibilité des groupes de travail, la prise en considération des caractéristiques de l'environnement permet dynamiquement et selon le type de tâche à réaliser, d'organiser la topologie efficacement. La succession des têtes de grappe forme un arbre hiérarchisé dans lequel les contraintes liées à l'environnement sont isolées. Leurs impacts sont locaux et n'affectent plus globalement l'avancement du traitement de la requête. La taille de ces grappes est optimisée selon le besoin nécessaire à la réalisation de la requête et il n'y a pas de gaspillage de ressource. Cette rétroaction de la topologie avec l'environnement permettra à notre architecture GRID de nouvelle génération de conserver sa performance dans un milieu dynamique ou mobile.

Afin de mettre en évidence les améliorations de performance apportées, nous devons mettre en pratique notre GRID et déterminer les coefficients du mécanisme de formation des groupes de travail. Pour y parvenir, le chapitre suivant présente l'implémentation et l'évaluation de notre nouvelle génération de GRID.

CHAPITRE 4
IMPLÉMENTATION ET ÉVALUATION

Suite à nos recherches, nous avons introduit une nouvelle génération de GRID dans le domaine des réseaux d'ordinateurs. Pour compléter cette proposition, nous devons tester et évaluer l'architecture GRID que nous proposons. Pour ce faire, nous avons choisi de développer un simulateur de GRID plutôt que l'intergiciel de notre GRID. L'outil de simulation de GRID nous permettra tout d'abord de tester et de valider le comportement global de l'architecture de notre GRID. Puisqu'il s'agit de simulation et non de cas réel, nous pouvons faire varier sans difficulté le nombre de ressources dans la GRID ainsi que le type d'environnement dans lesquels elles sont localisées. De plus, cet outil nous permettra de comparer les performances de notre architecture GRID avec celles des générations précédentes.

Ce chapitre expose dans une première partie le processus de développement suivi afin d'implémenter le simulateur de GRID. Pour chacune des générations de GRID, nous détaillerons les différents éléments et mécanismes de leur architecture qui seront implémentés dans le simulateur. Dans la deuxième partie du chapitre, nous évaluerons les différentes générations de GRID implémentées. Pour ce faire, nous mettrons l'emphase sur l'évaluation et l'analyse des performances de la formation des groupes de travail et du traitement de leurs requêtes.

4.1 Simulateur de GRID

Le simulateur de GRID est un outil important pour tester et évaluer l'architecture GRID que nous proposons. Il ne se limite pas seulement à simuler notre GRID, mais aussi toutes celles des générations précédentes. Le processus de son développement se doit d'être soigné afin d'obtenir des simulations proches de la réalité.

Tout d'abord, nous spécifierons les fonctionnalités qui devront être offertes par le simulateur. La période de nos recherches étant restreinte, nous devons prendre pour acquis certains éléments et mécanismes des GRID. Après avoir cité les suppositions que nous ferons dans nos simulations, nous présenterons l'architecture du simulateur de GRID, puis le choix de l'environnement de développement. Suite à cela, nous présenterons l'implémentation du simulateur et des GRID qui y seront simulées, suivie d'une brève exposition des activités de validation et de l'interface graphique du simulateur.

4.1.1 Spécifications

Le rôle de cet outil est clair, il doit permettre de simuler et d'évaluer les GRID des générations existantes en plus de celle que nous proposons. Pour clarifier les fonctionnalités qui doivent être offertes par le simulateur de GRID, nous annonçons la liste suivante des spécifications de cet outil :

1. Permettre de former un réseau d'ordinateurs :
   - 1.1. Définir la génération de la GRID.
   - 1.2. Choisir le nombre de ressources dans la GRID.
   - 1.3. Déterminer les connexions entre les ressources.
   - 1.4. Déterminer le type d'environnement.
   - 1.5. Visualiser le réseau d'ordinateurs :
     - 1.5.1. Représenter les nœuds de différentes couleurs selon leurs rôles.
     - 1.5.2. Représenter les liens entre les nœuds.
     - 1.5.3. Afficher la somme de la puissance de calcul de toutes les ressources.
   - 1.6. Permettre de sauvegarder la configuration en cours.
   - 1.7. Permettre de charger une ancienne configuration.
2. Permettre d'éditer une ressource :
   - 2.1. Déterminer la puissance de calcul.
   - 2.2. Déterminer l'espace disque disponible.
   - 2.3. Déterminer l'espace de la mémoire vive disponible.
   - 2.4. Déterminer l'autonomie restante.
   - 2.5. Visualiser l'indice de mobilité déterminé à l'aide de la relation (3.11) :
     - 2.5.1. Déterminer le délai de temps moyen avant une déconnexion.
     - 2.5.2. Déterminer la probabilité de reconnexion après une déconnexion.
   - 2.6. Visualiser l'indice de performance déterminé à l'aide de la relation (3.10) :
     - 2.6.1. Déterminer le coefficient de la puissance de calcul.
     - 2.6.2. Déterminer le coefficient de l'espace disque.
     - 2.6.3. Déterminer le coefficient de l'espace de la mémoire vive.
     - 2.6.4. Déterminer le coefficient de l'autonomie restante.
   - 2.7. Visualiser l'indice d'élection déterminé à l'aide de la relation (3.13) :
     - 2.7.1. Déterminer le coefficient de l'indice de performance.

2.7.2. Déterminer le coefficient de l'indice de mobilité.

2.7.3. Déterminer le coefficient de l'indice de qualité de service du réseau.

3. Permettre d'éditer un lien entre deux ressources :

3.1. Visualiser l'indice de la qualité de service déterminé à l'aide de la relation (3.12) :

3.1.1. Déterminer le débit de chaque sens de la connexion.

3.1.2. Déterminer le délai de réponse de chaque sens de la connexion.

3.1.3. Déterminer le coefficient du poids du débit.

3.1.4. Déterminer le coefficient du poids du délai de réponse.

3.2. Déterminer la fiabilité du lien.

4. Permettre la formation de groupe de travail selon la génération de la GRID :

4.1. Pour la première génération de GRID :

4.1.1. Déterminer l'hôte.

4.1.2. Former une GRID avec les voisins directs de l'hôte.

4.1.3. Former un groupe de travail de la taille de la GRID.

4.2. Pour la deuxième génération de GRID :

4.2.1. Déterminer l'hôte.

4.2.2. Former une GRID avec les voisins directs de l'hôte.

4.2.3. Former un groupe de travail de la taille de la GRID.

4.3. Pour la troisième génération de GRID :

4.3.1. Déterminer l'hôte.

4.3.2. Déterminer la qualité de service de la requête.

4.3.3. Former un groupe de travail :

4.3.3.1. Ne considérer que les voisins directs de l'hôte.

4.3.3.2. Respecter l'indice de qualité de service de la requête.

4.4. Pour la GRID proposée de nouvelle génération :

4.4.1. Déterminer l'hôte.

4.4.2. Déterminer la qualité de service de la requête.

4.4.3. Former un groupe de travail :

4.4.3.1. Considérer toutes les ressources de la GRID.

4.4.3.2. Hiérarchiser le groupe à l'aide de la stratégie de formation de groupe proposée.

4.4.3.3. Respecter l'indice de qualité de service de la requête.

5. Pouvoir soumettre une requête à traiter au groupe de travail :

5.1. Pour la première génération de GRID :

5.1.1. Déterminer la taille de la requête à traiter.

5.1.2. Déterminer le délai entre chaque étape de la simulation.

5.1.3. Déterminer un temps d'arrêt d'une simulation.

5.2. Pour la deuxième génération de GRID :

5.2.1. Déterminer la taille de la requête à traiter.

5.2.2. Déterminer le délai entre chaque étape de la simulation.

5.2.3. Déterminer un temps d'arrêt d'une simulation.

5.3. Pour la troisième génération de GRID :

5.3.1. Déterminer la taille de la requête à traiter.

5.3.2. Déterminer le délai entre chaque étape de la simulation.

5.3.3. Déterminer un temps d'arrêt d'une simulation.

5.3.4. Visualiser la probabilité moyenne de reconnexion des nœuds.

5.4. Pour la GRID proposée de nouvelle génération :

5.4.1. Déterminer la taille de la requête à traiter.

5.4.2. Déterminer le délai entre chaque étape de la simulation.

5.4.3. Déterminer un temps d'arrêt d'une simulation.

5.4.4. Visualiser la probabilité moyenne de reconnexion des nœuds.

5.4.5. Visualiser la probabilité moyenne de déconnexion douce des nœuds.

5.5. Permettre d'exporter des données de simulation :

5.5.1. Exporter sous le format Excel le temps de simulation.

5.5.2. Exporter sous le format Excel l'avancement du traitement global de la requête.

Le suivi de ces spécifications permet de réaliser l'outil de simulation dont nous avons besoin. Cependant, nous devons supposer certains mécanismes des GRID comme acquis, afin de ne pas avoir à implémenter tous les éléments qui ne sont pas nécessaires à l'évaluation des performances des différentes générations de GRID.

4.1.2 Suppositions

Les intergiciels de GRID sont des logiciels complexes, chacun d'eux est le fruit de plusieurs mois et années de recherche. Comme nous l'avons mentionné, nous devons prendre pour acquis les mécanismes qui n'ont pas de réel impact sur la performance général de la GRID. Pour chacune des générations, nous présentons l'ensemble des suppositions que nous avons faites :

A. Pour toutes les générations de GRID :

A.I. Nous tenons pour acquis les mécanismes de sécurité : La problématique de la sécurité dans les GRID a été résolue dans les GRID de troisième génération. Des mécanismes d'authentification, d'autorisation et de vérification ont été formulés dans la littérature. Notre priorité n'étant pas la sécurité, nous avons choisi de prendre ces mécanismes comme acquis. Il est évident que l'utilisation de GRID dans un autre domaine que celui de la recherche nécessite des mécanismes de sécurité robustes; cela afin d'éviter l'exploitation non autorisée de la GRID, le vol d'informations et l'altération des informations échangées.

A.II. Nous faisons abstraction des langages informatiques de programmation parallèle : Comme il s'agit d'une simulation et non d'un traitement réel d'une tâche, nous avons choisi de faire abstraction des langages de programmation des tâches. Sans code de programmation, nous définissons les requêtes à traiter comme un ensemble d'instructions informatiques à exécuter. C'est le nombre total d'instructions à traiter pour réaliser une requête qui spécifie sa taille dans la simulation. Pour faire un rapprochement avec la réalité, plus la taille d'une requête est grande dans la simulation, plus celle-ci devrait être complexe et longue à traiter par une GRID réelle.

A.III. Nous ignorons la coopération entre ressources : La coopération entre les ressources nécessite une gestion avancée des tâches de l'intergiciel des GRID. À cause du délai de temps réservé pour l'implémentation du simulateur et de l'architecture de notre GRID, nous n'avons pas pu

considérer le cas d'utilisation des GRID où les nœuds coopèrent afin de réaliser une tâche commune.

A.IV. Nous supposons que la fiabilité d'un lien représente la probabilité d'une déconnexion douce : En effet, si le nœud *n* ne se déconnecte pas de lui-même de manière douce, c'est que la connexion le liant au nœud *n'* a été coupée. Nous évaluons donc la probabilité d'une déconnexion forte à l'aide de la relation (4.1) :

$$\begin{aligned} P_{Deconnexion\_Forte_n} &= 1 - F_{Lien_{nn'}} \\ &= 1 - P_{Deconnexion\_Douce_n} \end{aligned} \tag{4.1}$$

Où :

- $P_{Deconnexion\_Forte_n}$ représente la probabilité d'une déconnexion forte du nœud *n*;
- $F_{Lien_{nn'}}$ représente la fiabilité du lien du nœud *n* à *n'*;
- $P_{Deconnexion\_Douce_n}$ représente la probabilité d'une déconnexion douce du nœud *n*.

A.V. Nous supposons que la GRID ne peut traiter plus d'une requête à la fois : Nous supposons qu'il n'est pas nécessaire d'offrir dans le simulateur les fonctionnalités nécessaires aux traitements de plusieurs requêtes en même temps par une même GRID. Évaluer le traitement d'une seule requête à la fois sera suffisant à notre analyse de performance.

A.VI. Nous supposons qu'une ressource informatique ne traite qu'une tâche à la fois: Afin de simplifier l'implémentation des intergiciels des GRID, nous ne considérons pas les traitements multitâches.

A.VII. Nous supposons que la puissance de calcul de 1GHz d'une ressource équivaut à 10^9 instructions informatiques pouvant être traitées à la seconde par celle-ci : Dépendamment de l'architecture et du modèle de l'unité de calcul, le nombre d'instructions informatiques pouvant être traité à la seconde varie. Afin de simplifier l'évaluation des GRID, nous supposons

pour toutes les unités de calcul qu'1GHz équivaut à 10^9 instructions informatiques par seconde.

A.VIII. Nous supposons qu'une instruction informatique est de 4 octets : Pour pouvoir évaluer le temps nécessaire au transfert d'une tâche dans une interconnexion, nous considérons que les instructions informatiques sont traitées par des processeurs de 32bits (8*4octets).

A.IX. Nous supposons qu'un message (autres qu'une tâche) échangé est de taille négligeable : Comme les messages échangés sont généralement des messages de signalisation courts dont la taille est difficile à évaluer, nous considérons qu'ils sont de tailles négligeables.

B. Pour les GRID de troisième génération :

B.I. Nous supposons qu'un seul groupe de travail peut être formé à la fois : Suite à la supposition A.V, le simulateur ne permet que de former un groupe de travail à la fois, bien que cette génération supporte la formation de plusieurs groupes de travail dans une même GRID.

B.II. Nous supposons que seuls les services de coordination et de partage de la puissance de calcul sont nécessaires à implémenter : Notre évaluation des GRID est axée sur leurs performances à traiter des requêtes. Nous supposons donc que seuls les services permettant de former un groupe de travail et de partager la puissance de calcul des ressources seront nécessaires pour comparer leurs performances.

C. Pour les GRID proposée de nouvelle génération :

C.I. Nous supposons qu'un seul groupe de travail peut être formé à la fois : Suite à la supposition A.V, le simulateur ne permet que de former un groupe de travail à la fois. Cela, bien que cette génération de GRID supporte aussi la formation de plusieurs groupes de travail.

C.II. Nous supposons que seuls les services de coopération et de partage de la puissance de calcul sont nécessaires à implémenter : Tout comme les GRID de troisième génération (voir supposition B.II), nous supposons que seuls les services permettant de former un groupe de travail et de partager la

puissance de calcul des ressources seront nécessaires pour comparer leurs performances.

C.III. Nous supposons que les GRID ne traitent qu'un seul type de requête : Le type application commun à toutes les générations de GRID est celui de la distribution des calculs intenses afin d'accélérer leurs réalisations. Suite à la supposition C.II, seules les requêtes de ce type seront nécessaires à implémenter afin d'évaluer la performance de leurs traitements.

Les spécifications et suppositions du simulateur de GRID étant formulées, nous poursuivons le processus de développement avec l'exposition de l'architecture de l'outil de simulation.

4.1.3 Architecture du simulateur de GRID

À l'aide de cet outil nous serons en mesure d'évaluer la performance des différentes générations de GRID et de les comparer à la nouvelle génération que nous proposons. Pour ce faire, nous devons nous éloigner légèrement de nos objectifs initiaux de recherche afin de proposer un système informatique simple et efficace qui nous permettra de simuler les différentes GRID.

Dans cette section, nous allons tout d'abord présenter les déférents éléments qui constituent l'architecture globale du simulateur. Nous identifierons les interactions entre ces éléments ainsi que le digramme UML des classes qui sera à implémenter.

4.1.3.1 Présentation de l'architecture globale

L'architecture du simulateur est constituée d'un ensemble d'éléments indépendants interagissant entre eux, la Figure 4.1 présente une vision globale du système. Nous y retrouvons les éléments suivants :

- Les libraires : Elles regroupent toutes les bibliothèques informatiques nécessaires à l'exécution de notre simulateur dans un système d'exploitation. De plus, pour visualiser la progression des étapes de la simulation, nous utiliserons des librairies graphiques. Grace à elles, nous pourrons offrir un aperçu de toutes les ressources de la GRID et de leurs fonctionnements.
- Les contrôleurs : Ils regroupent les intergiciels de GRID et le contrôle des outils de conception et de simulation. Il y a un intergiciel pour chacune des générations de GRID qui définit les mécanismes nécessaires à son fonctionnement. Le contrôle des

outils représentent les méthodes des fonctionnalités désirées qui sont exécutées lors des interactions entre l'utilisateur et le simulateur.

- Les interfaces : Elles regroupent tous les éléments qui rendent l'expérience d'utilisation du simulateur plus agréable. Elles constituent toutes les interfaces graphiques de la barre d'outils et de l'espace de simulation. La barre d'outils de conception et de simulation a été introduite afin de simplifier la saisie d'information ainsi que la manipulation des éléments de la simulation. L'espace de simulation représente le plan dans lequel une GRID sera représentée et où la simulation se déroulera.
- Les modèles : Ils regroupent les différents objets qui seront utilisés par le simulateur. Les principaux sont les nœuds, les liens, le réseau et les fichiers. Comme leurs noms l'indiquent, l'objet « nœud » représente une ressource de la GRID, l'objet « lien » représente l'interconnexion entre deux nœuds et l'objet « réseau » représente une GRID formée des objets de « nœud » et de « lien ». Quant à lui, l'objet « fichier » permet de sauvegarder et de charger une GRID dans le simulateur.

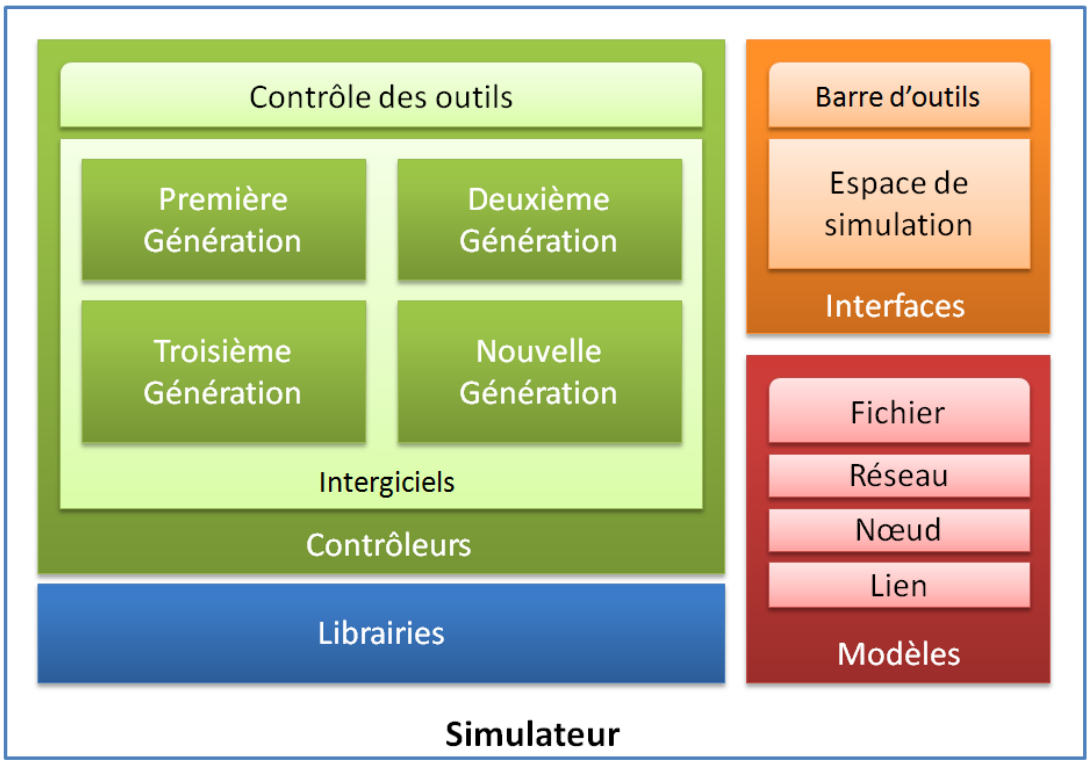


Figure 4.1 - Vision globale de l'architecture du simulateur

4.1.3.2 Interactions entre les éléments de l'architecture

Une simulation est le résultat de la coopération des différents éléments indépendants de l'architecture du simulateur. Tout commence par l'utilisation des outils de conceptions et de simulations par un usager du simulateur. Quand l'utilisateur interagit avec l'interface graphique, des événements sont envoyés aux contrôles des outils. Chaque outil fait appel à son propre contrôleur qui exécute les opérations désirées. Ces opérations visent à modifier le réseau d'ordinateurs simulé. Un réseau d'ordinateurs est lui-même composé d'un ensemble de nœuds et de liens. Les contrôleurs d'outils permettent aussi de choisir et de paramétrer l'intergiciel de la génération de GRID que nous voulons simuler. C'est l'ensemble de l'intergiciel d'une GRID et d'un réseau d'ordinateurs qui constitue l'espace de simulation. Cet espace de simulation est la représentation visuelle de la GRID; elle peut être rééditée par l'usager à l'aide des outils offerts. Ou encore, elle peut être sauvegardée dans un fichier ou aussi restaurée d'une sauvegarde précédente. La Figure 4.2 représente ce flux de donnée :

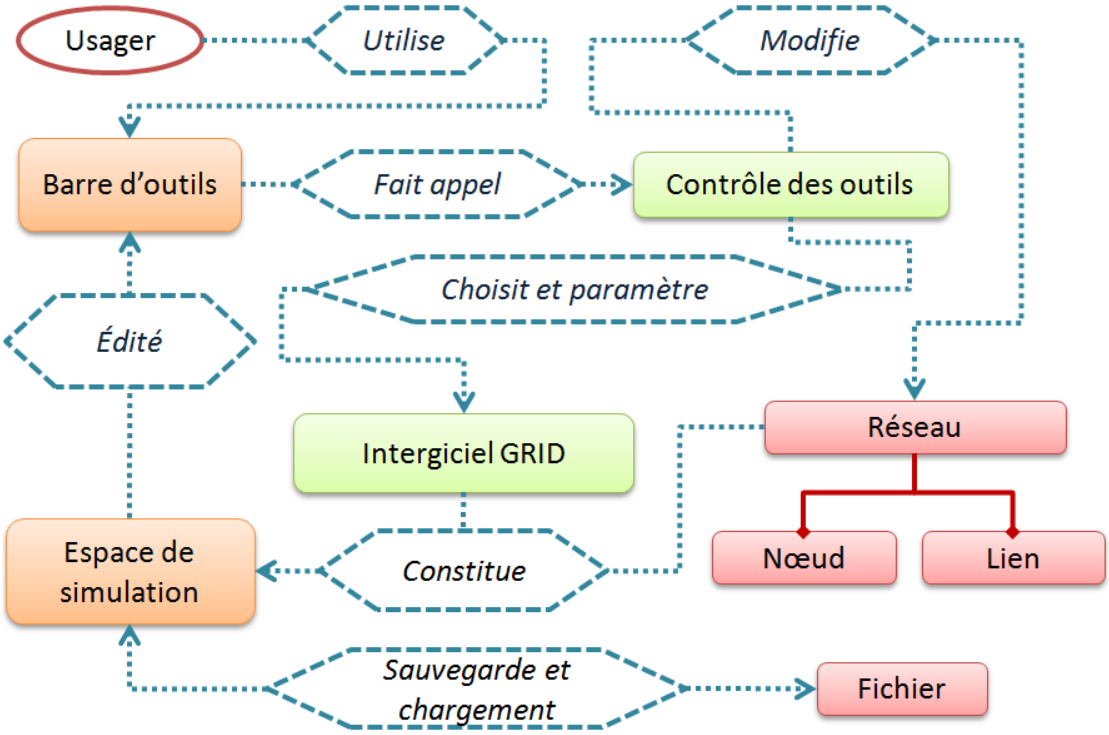


Figure 4.2 - Flux de données dans l'architecture du simulateur

Chaque élément de l'architecture du simulateur étant indépendant, le flux de données qui parcourt les éléments est simple et efficace. Cette division appropriée de l'architecture se reflète dans le diagramme de classe du simulateur.

4.1.3.3 Diagramme UML de classe

Pour compléter la présentation de l'architecture du simulateur, nous allons présenter le diagramme des classes qui seront à implémenter. Rappelons qu'une « classe » est une représentation informatique d'un objet ou d'un élément. Chaque classe est constituée de « méthodes » et d'« attributs ». Une méthode est une fonction pouvant être exécutée par la classe et un attribut, représentant un composant de celle-ci. Tout comme le flux de données, le diagramme UML de classe représente les interactions entre les différents éléments d'une architecture. Il est cependant davantage orienté vers le code informatique et sert de guide pour implémenter les différents éléments d'une architecture. Pour notre simulateur de GRID, nous proposons le diagramme UML de classe illustré à la Figure 4.3. L'architecture de notre simulateur de GRID est constituée de 12 classes et d'un ensemble de librairies graphiques :

- Classe « Outils » : Elle est composée de tous les boutons de la barre d'outils mise à disposition dans l'interface graphique du simulateur. Il y a un bouton pour chacune des fonctionnalités offertes à l'utilisateur, à savoir redessiner le réseau d'ordinateurs, former un groupe de travail dans la GRID, lancer la simulation du traitement d'une tâche, déplacer un nœud, ajouter un nœud, supprimer un nœud, créer un lien entre deux nœuds, supprimer un lien entre deux nœuds, sauvegarder la simulation, charger une simulation et finalement quitter le simulateur.
- Classe « Contrôleurs » : Elle associe à chaque bouton de l'interface une méthode. Quand un utilisateur utilise un outil, le contrôleur exécute la méthode permettant de réaliser l'opération demandée.
- Classe « Intergiciel » : C'est une classe « abstraite », c'est-à-dire que ses méthodes sont déclarées sans y être définies. Effectivement, chaque intergiciel de GRID dans notre simulateur devrait disposer d'un mécanisme pour former un groupe de travail ainsi que d'un autre pour traiter une tâche. Cependant, la définition de ces méthodes est propre à chaque génération de GRID. L'avantage d'utiliser ce modèle est de rendre interchangeable les classes d'intergiciels qui hériteront de cette classe abstraite selon la génération que nous souhaitons.
- Classe « 1G » : Représente l'intergiciel des GRID de première génération. Elle hérite de la classe « Intergiciel » et définit les mécanismes propres à cette génération.

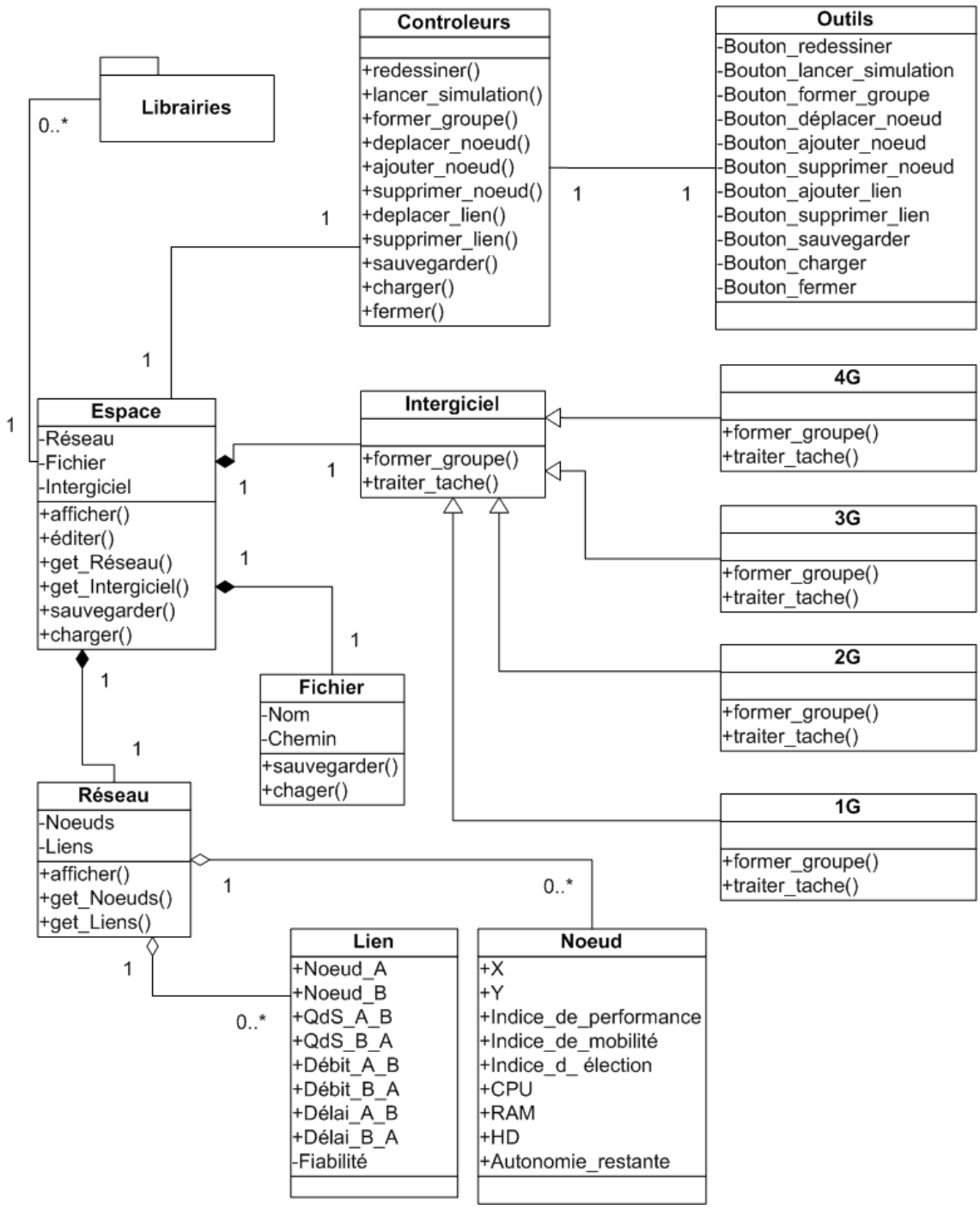


Figure 4.3 - Diagramme UML de classe de l'architecture du simulateur

- Classe « 2G » : Représente l'intergiciel des GRID de deuxième génération. Elle hérite de la classe « Intergiciel » et définit les mécanismes propres à cette génération.
- Classe « 3G » : Représente l'intergiciel des GRID de troisième génération. Elle hérite de la classe « Intergiciel » et définit les mécanismes propres à cette génération.

- Classe « 4G » : Représente l'intergiciel de l'architecture GRID de nouvelle génération que nous proposons. Elle hérite de la classe « Intergiciel » et définit les mécanismes que nous avons présentés dans le chapitre précédent.
- Classe « Fichier » : Cette classe définit un fichier dans lequel nous pouvons sauvegarder et charger des données en spécifiant son nom et sa localisation.
- Classe « Nœud » : Elle contient toutes les informations définissant une ressource informatique. Nous y retrouvons les coordonnées géométriques du nœud dans le plan où se déroulera la simulation, l'indice de performance, l'indice de mobilité, le dernier indice d'élection évalué, la puissance de calcul de son CPU, la taille disponible de sa mémoire vive, l'espace disque restant ainsi que son autonomie résiduelle.
- Classe « Lien » : Elle contient toutes les informations définissant une interconnexion entre les nœuds A et B du plan. Nous y trouvons l'indice de la qualité de service, le débit, le délai de réponse dans chaque sens et la fiabilité du lien. De plus, elle dispose de deux méthodes permettant d'envoyer des messages.
- Classe « Réseau » : Elle est formée d'un ensemble de nœuds et de liens et offre une méthode pour présenter le réseau dans un plan.
- Classe « Espace » : C'est la classe principale de notre simulateur, elle regroupe tous les éléments nécessaires à la réalisation d'une simulation, c'est-à-dire un réseau d'ordinateurs, un intergiciel de GRID et un fichier de sauvegarde. À l'aide des librairies graphiques, elle représente visuellement la GRID et les étapes de la simulation. Elle interagit avec la classe des contrôleurs afin de permettre à l'utilisateur de manipuler l'espace de simulation (voir Figure 4.2).

L'architecture du simulateur étant présentée, nous pouvons maintenant choisir les outils informatiques qui nous permettrons de réaliser notre simulateur de GRID.

4.1.4 Choix de l'environnement

Il existe une multitude d'outils et de langages informatiques permettant le développement d'applications. Parmi les plus populaires, nous avons choisi JAVA [25] comme langage de programmation car il offre les avantages suivants :

- ✓ c'est un langage de programmation qui est orienté « objet »;
- ✓ il incite à programmer des classes autonomes et réutilisables;

- ✓ il est indépendant du système d'exploitation;
- ✓ de nombreuses librairies open-source sont disponibles;
- ✓ il permet de gérer facilement les processus logiciels (*Threads*);
- ✓ la plateforme d'exécution et les outils de développement sont peu encombrants.

Comme outil de développement, nous avons choisi Netbeans IDE [26]. C'est un outil de développement entièrement en JAVA qui permet de développer des applications avec les langages informatiques populaires et émergents. Il n'y a pas d'incertitude technologique dans le développement de notre simulateur de GRID; JAVA et Netbeans sont des outils informatiques stables et matures. Leurs limitations sont connues et dans le cas d'un besoin d'aide, il existe une grande communauté virtuelle offrant des solutions aux problèmes les plus rencontrés.

4.1.5 Implémentation des GRID existantes

Après la réalisation des phases de création et d'élaboration dans un processus de développement, la phase d'implémentation peut débuter. Dans ce mémoire, nous n'allons pas présenter l'implémentation de toutes les classes de l'architecture du simulateur de GRID. Nous allons nous recentrer sur nos objectifs et mettre plutôt l'accent sur l'implémentation des différents mécanismes des intergiciels de GRID.

Dans cette section, pour chacune des générations de GRID existantes, nous allons présenter l'implémentation des deux méthodes de leurs classes.

4.1.5.1 Première et deuxième génération de GRID

Nous distinguons les différentes générations de GRID par la topologie de leurs ressources informatiques dans le réseau ainsi que par l'architecture de leurs intergiciels. Dans les deux premières générations de GRID, la topologie des ressources est la même. Elle se limite au serveur de coordination et ses ressources voisines. De plus, toutes les ressources de la GRID dont le serveur de coordination forment le groupe de travail.

La différence entre la première et la deuxième génération de GRID réside dans leurs capacités à traiter différents types de tâches. Les GRID de première génération sont dédiés à un seul type de tâches contrairement aux GRID de seconde génération. Cependant, nous avons fait abstraction des types de tâches dans nos simulations (voir proposition A.II dans la section 4.1.2). Puisque les requêtes soumises aux GRID et les sous-tâches qui en découlent ne sont plus que des

instructions informatiques à traiter, les intergiciels des deux premières générations de GRID à simuler sont identiques. Les méthodes des classes « 1G » et « 2G » sont donc similaires.

Pour implémenter les classes des intergiciels de première et seconde génération, nous définissions les méthodes issues des mécanismes de formation de groupe de travail et du traitement des tâches. Ces générations de GRID étant les plus simples à simuler, cette première présentation de l'implémentation d'intergiciel de GRID servira d'exemple initial avant d'aborder des générations plus complexes.

Formation du groupe de travail

Pour clarifier le mécanisme de formation du groupe de travail dans ces deux générations de GRID, nous allons présenter les différentes étapes de ce mécanisme dans les Tableau 4.1 et 4.2. Chaque étape de cette formation est accompagnée d'une description de celle-ci ainsi que d'une représentation graphique du réseau d'ordinateurs dans lequel la GRID opère.

Tout d'abord, l'étape initiale *0* est définie par la classe « réseau ». Comme nous l'avons vu dans le diagramme UML de classe à la Figure 4.3, c'est cette classe qui spécifie toutes les ressources et tous les liens du réseau d'ordinateurs.

Tableau 4.1 - Étapes de formation d'un groupe de travail dans une GRID 1G et 2G (1re partie)

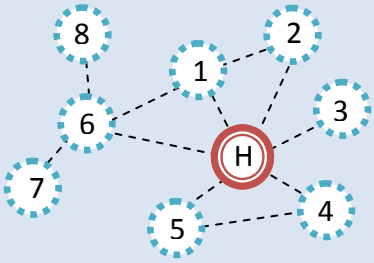
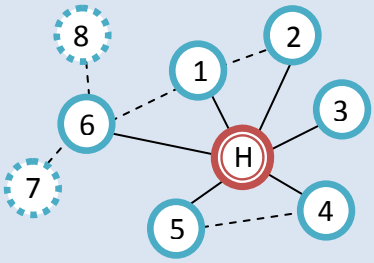
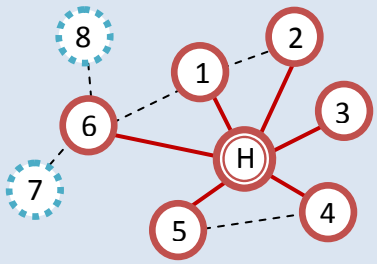
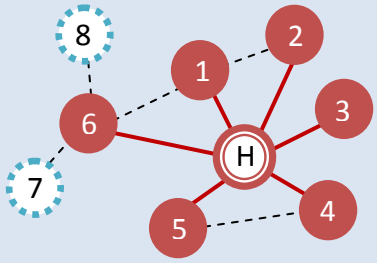
Étape	Description	Représentation
0	Un réseau d'ordinateurs est composé de 9 ressources informatiques. Ils sont interconnectés à l'aide de liens bidirectionnels. Lorsqu'il y a un lien entre deux ressources, elles sont alors capables de communiquer directement entre elles ; par exemple, la ressource (1) peut communiquer directement avec la (2), ou encore la (4) avec la (5). La ressource (H) représente le serveur de coordination.	
1	Les ressources tentent de se connecter au serveur de coordination. La position du serveur de coordination dans le réseau est connue de toutes les ressources. Seules les ressources (1), (2), (3), (4), (5) et (6) qui sont interconnectés au serveur de coordination peuvent signaler leurs présences. Elles forment le voisinage du serveur de coordination.	

Tableau 4.2 - Étapes de formation d'un groupe de travail dans une GRID 1G et 2G (2e partie)

Étape	Description	Représentation
2	Le serveur de coordination forme la GRID avec toutes les ressources voisines qui se sont connectées à lui. Nous pouvons faire remarquer que les ressources (7) et (8) qui n'ont pas de liens directs avec le serveur de coordination sont ignorées. Mais aussi, tous les liens du réseau d'ordinateurs ne passant pas par le serveur de coordination resteront inutilisés.	
3	Lorsque le serveur de coordination reçoit une requête à traiter, il fait appel à toutes les ressources de sa GRID pour former un groupe de travail et commencer le traitement de la tâche.	

Il faut trois étapes au serveur de coordination pour former un groupe de travail. Dans ses deux premières générations de GRID, l'intergiciel du serveur de coordination est différent de celui des ressources. Pour chacune des étapes *1*, *2* et *3* de la formation du groupe de travail, et selon le rôle de la ressource informatique dans la GRID, les Tableaux 4.3 et 4.4 spécifient le pseudo-code de l'implémentation de la méthode *former_groupe* des classes « 1G » et « 2G ».

Tableau 4.3 - Pseudo-code de la méthode former_groupe d'une GRID 1G et 2G (1re partie)

Étape	Rôle	Pseudo-code
1	Serveur de coordination	▪ Tant *qu'il n'y a aucune requête reçue* : • Si *un message de signalisation est reçu* : o Répondre en confirmant la réception. o Ajouter la ressource qui a envoyé le message de signalisation dans la liste des ressources voisines.
	Ressource	▪ Tant *que le serveur de coordination n'a pas répondu* : • À *chaque minute* : o Pour *chaque lien* : … *(Suite au Tableau 4.4)*

Tableau 4.4 - Pseudo-code de la méthode *former_groupe* d'une GRID 1G et 2G (2e partie)

Étape	Rôle	Pseudo-code
1	(Suite) Ressource	♦ Envoyer un message de signalisation destiné au serveur de coordination. La ressource y indique sa puissance de calcul.
2	Serveur de coordination	▪ Pour *chaque ressource voisine* : • À *chaque minute* : o Envoyer un message de signalisation. o Si la ressource voisine ne répond pas, elle est retirée de la liste des voisins.
	Ressource	▪ Si *la ressource reçoit une confirmation de réception du message de signalisation qu'elle a envoyé précédemment* : • Arrêter l'envoi des messages de signalisation. • Répondre aux nouveaux messages de signalisation envoyés par le serveur de coordination. • Attendre la formation de la GRID.
3	Serveur de coordination	▪ Si *une requête à traiter est reçue* : • Pour *chaque ressource voisine* : o Confirmer son appartenance à la GRID. o Préparer les sous-tâches à envoyer.
	Ressource	▪ Si la ressource reçoit une confirmation d'appartenance à la GRID du serveur de coordination : • Préparer la ressource à la réception de tâche. • Attendre les sous-tâches à traiter.

À la fin de l'exécution de ce mécanisme, le serveur de coordination est prêt à soumettre des tâches aux ressources de la GRID qui sont en attente.

Traitement des tâches

Une fois le groupe de travail formé, la méthode *traiter_tâche* permet à la GRID de traiter une requête soumise. Tout comme la méthode précédente, il y a deux différentes versions de

l'implémentation de cette méthode. Il y a une version pour l'intergiciel du serveur de coordination et une autre pour celui des autres ressources de la GRID.

Dans ses deux premières générations, la simulation des traitements de tâches est assez simple. Le serveur de coordination et les ressources se partagent le nombre total d'instructions informatiques formant la requête proportionnellement à leurs puissances de calcul. Pour ce faire, nous utilisons la relation (4.2) :

$$Taille_{SousTache_n} = \frac{I_{Performance_{CPU_n}} \times N_{operations}}{\sum_{x=1}^{v} I_{Performance_{CPU_{n_x}}}} \quad (4.2)$$

Où :

- $Taille_{SousTache_n}$ représente le nombre d'instructions informatiques formant la sous-tâche à traiter par le nœud *n*;
- $I_{Performance_{CPU_n}}$ représente la puissance de calcul en GHz du nœud *n*;
- $N_{operations}$ représente le nombre total d'instructions informatiques nécessaires pour traiter la requête;
- $\sum_{x=1}^{v} I_{Performance_{CPU_{n_x}}}$ représente la somme de la puissance de calcul des *v* ressources $n_1, n_2, n_3 \ldots n_v$ de la GRID dont le serveur de coordination.

Dans les GRID réelles, le serveur de coordination ne participe pas au traitement des tâches. Il se concentre sur la division de la requête en sous-tâche et à coordonner l'avancement des traitements. Cependant, même si le serveur de coordination ne fait pas avancer directement le traitement de la requête, il exécute un grand nombre d'instructions informatiques nécessaires à la synchronisation et à la coordination des sous-tâches. Comme nous avons supposé dans notre simulation (voir proposition A.II dans la section 4.1.2) qu'une requête définit le nombre total d'instructions informatiques à exécuter par toute la GRID, nous considérons donc aussi le serveur de coordination dans la répartition des instructions informatiques au groupe de travail.

Suite à la déconnexion d'une ressource, le serveur de coordination redistribue les instructions informatiques de toutes les tâches qui ont été assignées à la ressource qui s'est déconnectée. Il ne prend en considération que les ressources restantes de sa GRID. La répartition des instructions informatiques se fait de la même manière que pour celle d'une requête. Chaque ressource restante

ajoutera les instructions de ses nouvelles sous-tâches à celles qu'elles ont déjà à traiter. Si toutes les ressources se déconnectent de la GRID, le traitement de la requête échoue.

Pour clarifier les différentes étapes de ce mécanisme de traitement des tâches, les Tableaux 4.5 et 4.6 spécifient à chaque étape le pseudo-code de l'implémentation de la méthode *traiter_tâche* des classes « 1G » et « 2G ».

Tableau 4.5 - Pseudo-code de la méthode *traiter_tâche* d'une GRID 1G et 2G (1re partie)

Étape	Rôle	Pseudo-code
1	**Description :** Le serveur de coordination reçoit une requête à traiter. Il divise le nombre d'instructions informatiques à traiter en autant de sous-tâches qu'il y a de ressources informatiques dans GRID.	
	Serveur de coordination	▪ Tout en poursuivant l'envoi des messages de signalisation : • Déterminer à l'aide de l'équation 4.1 et de sa puissance de calcul, le nombre d'instructions informatiques de sa sous-tâche à traiter. • Pour *chacune des ressources voisines de sa GRID* : o Déterminer à l'aide de l'équation 4.1 et de l'indication sur sa puissance de calcul reçue dans son message de signalisation, le nombre d'instructions informatiques de la sous-tâche à traiter par cette ressource. • Envoyer la taille calculée de sa sous-tâche à la ressource.
	Ressource	▪ Tout en répondant aux messages de signalisation : • Attendre le nombre d'instructions informatiques à traiter dans la sous-tâche qui lui est assignée par le serveur de coordination.
2	**Description :** Les ressources et le serveur de coordination traitent leurs sous-tâches. Si une ressource ne répond plus aux messages de signalisation, le serveur de coordination redistribue sa tâche à toute la GRID. Quand toutes les instructions informatiques sont réalisées, les ressources signalent la fin du traitement de leurs sous-tâches au serveur de coordination.	
	Serveur de coordination	▪ En parallèle : • Envoyer des messages de signalisation aux ressources. • Tant que le serveur de coordination n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche : o *À chaque seconde* : … *(Suite au Tableau 4.6)*

Tableau 4.6 - Pseudo-code de la méthode *traiter_tâche* d'une GRID 1G et 2G (3e partie)

Étape	Rôle	Pseudo-code
2	(Suite) Serveur de coordination	♦ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. • Tant *que toutes les ressources n'ont pas signalé la fin de leurs traitements* : o Si *une ressource ne répond pas à un message de signalisation* : ♦ La ressource est retirée du voisinage. ♦ Pour *chaque ressource restante de la GRID* : ◊ Déterminer à l'aide de l'équation 4.1 et de la puissance de calcul, le nombre d'instructions informatiques de la tâche de la ressource déconnectée qui sont à redistribuer à cette ressource. ◊ Envoyer à la ressource sa nouvelle sous-tâche à traiter. o Écouter les messages de fin de traitement provenant des ressources.
	Ressource	▪ En parallèle : • Répondre aux messages de signalisation. • Si *la ressource reçoit une sous-tâche* : o Tant *que la ressource n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche* : ♦ À *chaque seconde* : ◊ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. o Envoyer un message au serveur de coordination pour signaler la fin du traitement. • Si *la ressource se déconnecte du groupe de travail* : o Arrêter le traitement de la tâche en cours. o Effacer toutes les tâches en attentes.

Le pseudo-code des méthodes *former_groupe* et *traiter_tâche* des classes « 1G » et « 2G » étant exposé, les simulations des GRID de première et deuxième génération sont maintenant possibles dans notre outil de simulation de GRID.

4.1.5.2 Troisième génération de GRID

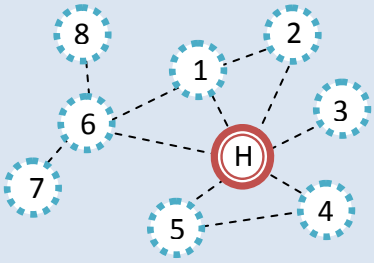
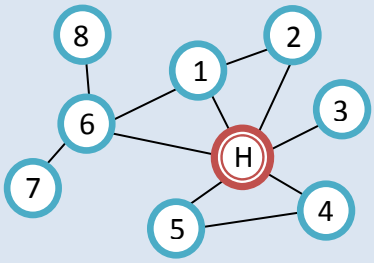
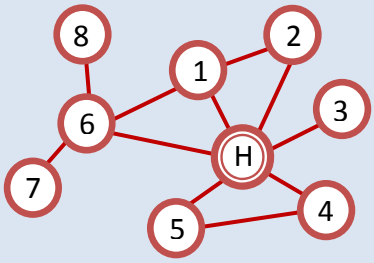
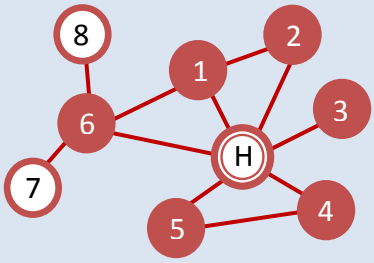
Nous poursuivons la phase d'implémentation avec la troisième génération de GRID. Des améliorations ont été faites dans l'architecture de l'intergiciel et la topologie des ressources informatiques. Dans les GRID de troisième génération, nous considérons toutes les ressources du réseau : elles forment les nœuds de la GRID. Le serveur de coordination est remplacé par un nœud hôte qui dispose d'un service de coordination. Nous pouvons donc avoir autant de groupes de travail dans une GRID que de nœuds ayant le service de coordination. L'intergiciel de tous les nœuds de la GRID est identique, seuls les différents services qu'ils offrent définissent leurs rôles dans celle-ci.

Formation du groupe de travail

Même si les GRID de troisième topologie sont formées par toutes les ressources du réseau, un groupe de travail ne peut s'étendre qu'aux nœuds voisins de l'hôte. La topologie du groupe de travail de cette génération est donc proche de celles des générations précédentes. Cependant, elle peut être modifiée pour assurer une certaine qualité de service. En effet, dans cette génération, une requête peut spécifier la qualité de service avec laquelle un groupe de travail devrait la traiter. La valeur de la qualité de service d'une requête représente la performance recommandée du groupe de travail afin de finaliser le traitement de la requête dans un délai respectable. La performance d'un groupe de travail est évaluée par la somme de la puissance de calcul des nœuds qui forment ce groupe de travail. Elle est mesurée en nombre d'instructions informatiques devant être effectuées en une seconde par le groupe de travail. Selon donc cette valeur, l'hôte adapte la topologie de son groupe de travail afin de répondre au besoin de performance.

Pour éclaircir cette nouvelle notion de qualité de service, nous allons présenter les différentes étapes du mécanisme de formation d'un groupe de travail des GRID de troisième génération dans le Tableau 4.7.

Tableau 4.7 – Étapes de formation d'un groupe de travail dans une GRID 3G

Étape	Description	Représentation
0	Soit le réseau d'ordinateurs détaillé dans l'étape initiale *0* du Tableau 4.1. La ressource (H) représente l'hôte.	
1	À l'aide d'échange de messages P2P, toutes les nœuds signalent leur présence à leurs voisins. Toutes les interconnexions sont utilisées et des messages peuvent être échangés entre tous les nœuds du réseau.	
2	Tous les nœuds du réseau forment la GRID. Chaque nœud connaît la position de tous les autres nœuds, leurs puissances de calcul, ainsi que les services qu'ils offrent.	
3	Lorsque l'hôte reçoit une requête à traiter avec une qualité de service spécifiée, il forme un groupe de travail dont la performance est légèrement supérieure à la valeur de la qualité de service de la requête. Si cette valeur peut être atteinte, l'hôte ne forme un groupe de travail qu'avec certains de ses nœuds voisins, dans le cas présent tous ces nœuds voisins sont réquisitionnés. Nous pouvons remarquer que les nœuds (7) et (8) font toujours partie de la GRID, mais ils ne peuvent pas être exploités par l'hôte.	

Tout comme les générations précédentes, trois étapes sont nécessaires pour former un groupe de travail. Les étapes *1* et *2* sont indépendantes des services offerts par les nœuds. En effet, même si un nœud de la GRID n'offre aucun service, il dispose de toutes les fonctionnalités pour réaliser

les deux premières étapes de la formation d'un groupe de travail. Il s'agit en fait des étapes de signalisation et de découverte d'autres nœuds de la GRID. Ce n'est qu`à dans la troisième étape que les services spécifient le rôle à jouer du nœud dans le groupe de travail et dans la GRID.

Pour chacune des étapes *1*, *2* et *3* présentées et selon les services offerts par les nœuds, les Tableaux 4.8 et 4.9 spécifient le pseudo-code de l'implémentation de la méthode *former_groupe* de la classe « 3G ».

Tableau 4.8 - Pseudo-code de la méthode *former_groupe* d'une GRID 3G (1re partie)

Étape	Service	Pseudo-code
1	Pour tous les nœuds	▪ En parallèle : • À *chaque minute* : o Pour *chaque lien* : ♦ Envoyer un message de signalisation. Le nœud y indique sa puissance de calcul et la liste de ses nœuds voisins. • Si *un message de signalisation est reçu* : o Répondre au message en y indiquant sa puissance de calcul et la liste de ses nœuds voisins. o Si *le nœud n'est pas déjà dans la liste des voisins* : ♦ Ajouter le nœud qui a envoyé le message de signalisation dans la liste des nœuds voisins.
2	Pour tous les nœuds	▪ En parallèle : • À *chaque minute* : o Pour *chaque lien*, le nœud continu à envoyer et à répondre aux messages définis dans l'étape précédente. • Pour *toutes les listes de voisinage reçues* : o Mettre à jour sa table de routage : ♦ Déterminer les chemins pour communiquer avec les nœuds qui ne sont pas des voisins. o Établir la liste des nœuds de la GRID et de leurs puissances de calcul. • Retransmettre les messages reçus qui sont destinés à un autre nœud de la GRID à l'aide de la table de routage.

Tableau 4.9 - Pseudo-code de la méthode *former_groupe* d'une GRID 3G (2e partie)

Étape	Service	Pseudo-code
3	Service de coordination	▪ En parallèle : • Poursuivre les procédures de l'étape 2. • Si *une requête à traiter est reçue* : o Si *une qualité de service de la requête est spécifiée* : ♦ Tant *que la performance du groupe de travail n'est pas supérieure à la valeur de la qualité de service de la requête* : ◊ Ajouter un nœud voisin au groupe de travail. o Sinon : ♦ Former un groupe de travail avec tous les nœuds voisins de l'hôte. ♦ Préparer les sous-tâches à envoyer.
	Partage de la puissance de calcul	▪ En parallèle : • Poursuivre les procédures de l'étape 2. • Si le nœud fait partie d'un groupe de travail : o Préparer le nœud à la réception de tâche. o Attendre les sous-tâches à traiter.

À la fin de l'exécution de la méthode *former_groupe*, un groupe de travail s'est formé autour de l'hôte ayant reçu la requête.

Traitement des tâches

Le traitement d'une requête par un groupe de travail d'une GRID de troisième génération est similaire à celui des GRID de première et seconde génération. Les sous-tâches sont réparties aux différents nœuds selon leurs puissances de calcul. La division de leurs charges est évaluée à l'aide de la relation (4.2) de la méthode *traiter_tâche* des classes « 1G » et « 2G ». Cependant, dans cette génération, durant le traitement de groupe de travail, si un nouveau nœud s'ajoute au voisinage de l'hôte, il peut être ajouté dynamiquement à son groupe de travail. Sa venue allégera la charge de travail des autres nœuds du groupe de travail en redistribuant le nombre des instructions informatiques restant à traiter à tous les nœuds du groupe de travail. Le choix de

l'ajout dynamique de nouveaux nœuds est pris par l'hôte selon l'équilibre entre la qualité de service de la requête et la performance du groupe de travail.

Comme nous l'avons fait pour les deux générations de GRID précédentes, nous considérons aussi l'hôte dans la division du nombre total d'instructions informatiques de la requête à soumettre aux ressources de la GRID. Dans les GRID réelles de cette génération, si l'hôte dispose aussi du service de partage de ressource, il considère automatiquement sa puissance de calcul dans la performance du groupe de travail.

Pour clarifier les différentes étapes de ce mécanisme de traitement des tâches, les Tableaux 4.10 à 4.12 spécifient à chaque étape, le pseudo-code de l'implémentation de la méthode *traiter_tâche* de la classe « 3G ».

Tableau 4.10 - Pseudo-code de la méthode *traiter_tâche* d'une GRID 3G (1re partie)

<table>
<tr><th>Étape</th><th>Service</th><th>Pseudo-code</th></tr>
<tr><td>Toutes</td><td>Pour tous les nœuds</td><td>▪ Poursuivre la mise à jour des nœuds définit l'étape 2 du Tableau 4.8.</td></tr>
<tr><td rowspan="3">1</td><td colspan="2">Description : L'hôte reçoit une requête à traiter. Il divise le nombre total d'instructions informatiques à traiter en autant de sous-tâches qu'il y a de nœuds dans son groupe de travail.</td></tr>
<tr><td>Service de coordination</td><td>▪ Déterminer à l'aide de la relation (4.1) et de sa puissance de calcul, le nombre d'instructions informatiques de sa sous-tâche à traiter.
▪ Pour chaque nœud de son groupe de travail :
• Déterminer à l'aide de la relation (4.1) et de la puissance de calcul du nœud, le nombre d'instructions informatiques de la sous-tâche à traiter par ce nœud.
• Envoyer au nœud sa sous-tâche à traiter.</td></tr>
<tr><td>Partage de la puissance de calcul</td><td>▪ Attendre qu'une sous-tâche soit assignée par l'hôte.</td></tr>
</table>

Tableau 4.11 - Pseudo-code de la méthode *traiter_tâche* d'une GRID 3G (2e partie)

Étape	Service	Pseudo-code
2		**Description :** Les nœuds traitent leurs sous-tâches. Si un nœud ne répond plus aux messages de signalisation, l'hôte redistribue sa tâche à tout son groupe de travail. S'il y a une nouvelle ressource voisine de l'hôte et que la performance de la GRID est inférieure à la valeur de la qualité de service de la requête, l'hôte ajoute le nouveau nœud au groupe de travail. Quand toutes les instructions informatiques sont réalisées, les nœuds signalent la fin du traitement de leurs sous-tâches à l'hôte.
	Service de coordination	▪ En parallèle : • Tant *que l'hôte n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche* : o À *chaque seconde* : ♦ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. • Si *un nœud ne répond pas à un message de signalisation* : o Le nœud est retiré de la liste de voisinage. o Pour *tous les nœuds restants du groupe de travail* : ♦ Déterminer à l'aide de la relation (4.1) et de la puissance de calcul, le nombre d'instructions informatiques de la tâche du nœud déconnectée qui sont à redistribuer à ce nœud. ♦ Envoyer sa nouvelle sous-tâche à traiter. • Si *l'hôte a un nouveau voisin* : o Si *la qualité de service de la requête n'est pas définie* : Ou o Si *elle est définie et est supérieure à la performance du groupe de travail* : Alors ♦ Ajouter le nouveau nœud à son groupe de travail. ♦ Demander aux nœuds du groupe de travail le nombre d'instructions informatiques qu'ils ont réalisées. ♦ Évaluer le nombre total d'instructions informatiques restant à traiter par le groupe de travail. ♦ … *(Suite au Tableau 4.12)*

Tableau 4.12 - Pseudo-code de la méthode *traiter_tâche* d'une GRID 3G (3e partie)

Étape	Service	Pseudo-code
2	(Suite) Service de coordination	♦ Pour *tous les nœuds du groupe de travail* : ◊ Envoyer un message d'arrêt du traitement en cours. ◊ Déterminer à l'aide de la relation (4.1) et de la puissance de calcul du nœud, le nombre d'instructions informatiques restant à faire traiter par ce nœud. ◊ Envoyer la nouvelle sous-tâche à traiter au nœud. • Tant *que tous les nœuds n'ont pas signalé la fin de leurs traitements* : o Écouter les messages provenant des nœuds.
	Partage de la puissance de calcul	▪ Tant *que le nœud est dans un groupe de travail*, en parallèle : • Si *le nœud reçoit une sous-tâche* : o Tant *que le nœud n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche* : ♦ À *chaque seconde* : ◊ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. o Envoyer un message à l'hôte pour signaler la fin du traitement. • Si *le nœud reçoit un message d'arrêt de traitement.* Ou • Si *le nœud se déconnecte du groupe de travail.* Alors o Arrêter le traitement de la tâche en cours. o Effacer toutes les tâches qui sont à traiter.

Le pseudo-code des méthodes *former_groupe* et *traiter_tâche* de la classe « 3G », nous permet d'implémenter l'intergiciel de la troisième génération de GRID.

4.1.6 Implémentation de l'intergiciel de notre architecture GRID

Dans notre nouvelle génération de GRID, nous nous sommes proposé de poursuivre l'élan d'amélioration de l'architecture de l'intergiciel des GRID de troisième génération, mais aussi de définir une topologie décentralisée plus performante des ressources informatiques. Tout comme la génération précédente : la GRID s'étend à tous les nœuds du réseau d'ordinateurs à l'aide de messages de signalisation P2P, une qualité de service de la requête peut-être spécifiée et l'intergiciel des nœuds est identique pour toute la GRID.

Bien qu'il existe un grand nombre de similarités entre les différents éléments de l'architecture de notre intergiciel et celui des GRID de troisième génération, des modifications majeures ont été apportées dans les mécanismes de formation d'un groupe de travail et de traitement d'une requête.

4.1.6.1 Formation d'un groupe de travail

Afin de rendre la topologie d'un groupe de travail plus flexible et de l'adaptée aux différents environnements des réseaux d'ordinateurs, nous avons introduit des concepts de mobilité dans l'architecture de l'intergiciel des nœuds, mais aussi équipé tous les nœuds de la GRID du service de coordination.

Dans notre proposition, un groupe de travail peut exploiter toutes les ressources informatiques d'une GRID et cela même si tous les nœuds qui la forment ne sont pas voisins. Pour y parvenir, nous divisons un groupe de travail en un ensemble de grappes hiérarchisées :

- chaque grappe est formée par un regroupement de nœuds autour d'un nœud hôte nommé « tête de grappe »;
- le choix des têtes de grappe est fait à l'aide d'un indice d'élection déterminé à partir de son indice de performance, de mobilité et de qualité de service de ses interconnexions;
- la hiérarchie des grappes dans un groupe de travail est représentée par le chemin des têtes de grappe partant du nœud ayant reçu la requête à traiter jusqu'au nœud le plus éloigné de la GRID.

Ce mécanisme est détaillé dans la section 3.3.3.2 du chapitre 3 portant sur la présentation de notre architecture. Pour simplifier et clarifier les étapes de la formation du groupe de travail

détaillé, nous posons les valeurs suivantes des termes de la relation (3.13) permettant de déterminer l'indice d'élection d'un nœud *n* du groupe de travail :

- $P(n) = 1$ et $C_P = 1$.
- Pour tous les voisins *t* de *n*, $Q(n,t) = 1$ et $C_Q = 0$.
- $M(n) = 0$ et $C_M = 0$.

Les Tableaux 4.13 à 4.15 illustrent la formation d'un groupe de travail dans une GRID de nouvelle génération.

Tableau 4.13 - Étapes de formation d'un groupe de travail dans une GRID 4G (1re partie)

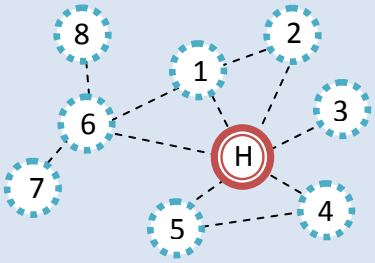
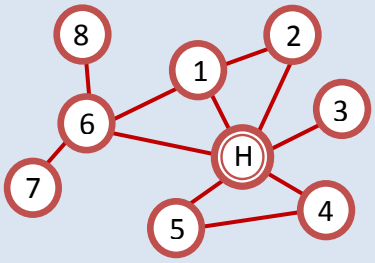
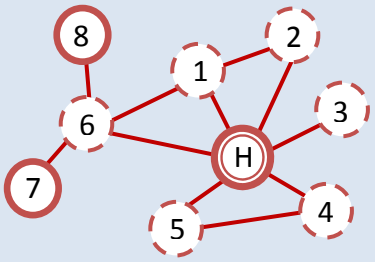
Étape	Description	Représentation
0	Soit le réseau d'ordinateurs détaillé dans l'étape initiale *0* du tableau 4.1. La ressource (H) représente le nœud qui recevra la requête à traiter; il offre le service de partage de sa puissance de calcul aussi.	
1	À l'aide d'échange de messages P2P, tous les nœuds signalent leur présence à leurs voisins. Toutes les interconnexions sont utilisées et des messages peuvent être échangés entre tous les nœuds du réseau. Tous les nœuds du réseau forment la GRID. Chaque nœud connaît la position de tous les autres nœuds, leurs indices de performance ainsi que les services qu'ils offrent. Ils évaluent aussi la qualité de service de leurs liens.	
2	Lorsque le nœud H reçoit une requête, il demande à ses nœuds voisins 1, 2, 3, 4, 5 et 6 d'évaluer leurs indices d'élection. À l'aide de la relation (3.13) et des valeurs des termes *P*, *Q*, *M*, C_P, C_Q *et* C_M posées pour simplifier cette présentation, les nœuds évaluent leurs indices d'élections : E(1) = 4, E(2) = 3, E(3) = 2, E(4) = 3, E(5) = 3, E(6) = 5 et E(H) = 7. Expliquons le calcul de E(6) : … *(Suite au Tableau 4.14)*	

Tableau 4.14 - Étapes de formation d'un groupe de travail dans une GRID 4G (2e partie)

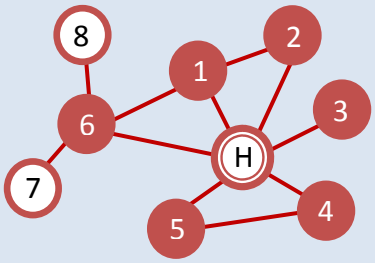
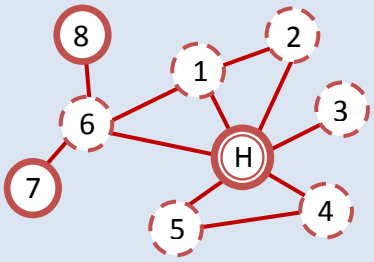
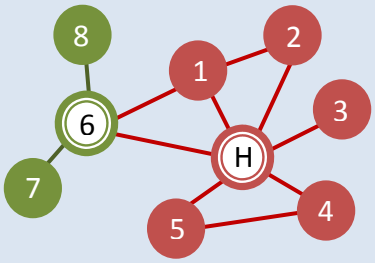
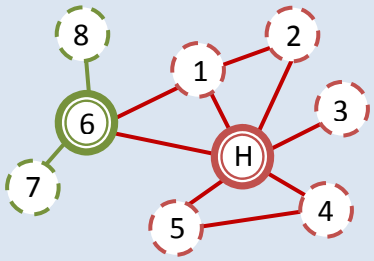
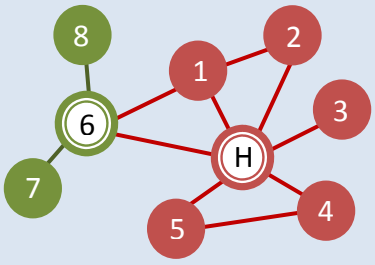
Étape	Description	Représentation
(Suite) 2	… pour déterminer son indice d'élection, le nœud 6 fait la somme des indices de performance de ses voisins ne faisant pas encore partie du groupe de travail et l'ajoute au sien. Dans la représentation graphique du réseau d'ordinateurs de la GRID à droite, le nœud 6 a pour voisin les nœuds 1, H, 7 et 8. Comme nous avons posé que *P(n)* = 1 pour tous les nœuds *n* de la GRID, alors la somme de l'indice de performance de ses nœuds voisins est égal à la valeur 4. À cela, le nœud 6 y ajoute son indice de performance P(6) afin d'obtenir son indice d'élection E(6)= 5.	*(Voir la représentation de l'étape 2 dans le Tableau 4.13)*
3	Le nœud H se choisit comme tête de grappe, car il a l'indice d'élection le plus élevé. Comme il n'y a pas de qualité de service de la requête de spécifiée dans celle-ci, il signale à tous ses voisins qu'ils font partie de sa grappe. Dans le cas contraire, il n'aurait choisi que les nœuds qui sont nécessaires pour obtenir une performance du groupe de travail qui soit satisfaisante. Cette première grappe forme la grappe principale du groupe de travail.	
4	Pour continuer à étendre le groupe de travail à l'ensemble des ressources informatiques du réseau, la tête de grappe H demande à ses nœuds voisins de réévaluer leurs indices d'élection. Il obtient : E(1) = 0, E(2) = 0, E(3) = 0, E(4) = 0, E(5) = 0, E(6) = 2. À part le nœud 6, tous les autres voisins de la tête de grappe H n'ont plus que des voisins faisant partie du groupe de travail. De plus, ils ne considèrent plus leurs propres indices de performance dans le calcul de leurs indices d'élection car ils font déjà partie du groupe de travail. Ils ont donc des indices d'élection nuls. Seul le nœud 6 a pour voisin les nœuds 7 et 8 qui ne font pas encore partie du groupe de travail, donc E(6) = 2.	

Tableau 4.15 - Étapes de formation d'un groupe de travail dans une GRID 4G (3e partie)

Étape	Description	Représentation
5	La tête de grappe H choisit le nœud 6 comme seconde tête de grappe. Comme il n'y a pas de qualité de service de la requête de spécifiée dans celle-ci, le nœud 6 signale à tous ses voisins ne faisant pas partie du groupe de travail, qu'ils font dorénavant partie de sa grappe. Dans le cas contraire, il n'aurait choisi que les nœuds voisins qui sont nécessaires pour obtenir une performance de groupe du travail qui soit satisfaisante à la qualité de service de la requête.	
6	Pour continuer à étendre le groupe de travail à l'ensemble des ressources informatiques du réseau, la tête de grappe H demande à ses nœuds voisins de réévaluer leurs indices d'élection. Il obtient : E(1) = 0, E(2) = 0, E(3) = 0, E(4) = 0, E(5) = 0. Le nœud 6 étant déjà une tête de grappe, il n'est plus à élire. Cependant, en tant que tête de grappe secondaire placée en dessous de la tête de grappe principale H selon la hiérarchie, il se doit de retransmettre la demande de calcul d'indice d'élection aux nœuds de sa grappe. La tête de grappe 6 reçoit donc E(7) = 0 et E(8) = 0; ces indices étant nuls, elle n'a pas à les réévaluer à l'aide de la relation (3.14) avant de présenter le meilleur des deux à la tête de grappe principale H. Dans le cas où il y a plusieurs nœuds avec la meilleure valeur de l'indice d'élection, c'est le premier nœud à avoir répondu qui est retenu.	
7	Tous les indices d'élection étant nuls, le groupe de travail ne peut plus s'étendre. La tête de grappe H lance le traitement des tâches.	

Dans cette nouvelle génération, le nombre d'étapes menant à la formation d'un groupe de travail n'est pas fixe. Moins il y a de ressources informatiques voisines dans la GRID, plus il faudra d'étapes au groupe de travail pour s'étendre à toutes celles-ci.

Ce mécanisme étant déjà présenté sous forme de pseudo-code dans la section 3.3.3.2, nous ne présentons pas le pseudo-code de la méthode *former_groupe* de la classe « 4G ».

4.1.6.2 Traitement d'une tâche

Les objectifs de performance de notre architecture GRID nous ont poussés à décentraliser le traitement des requêtes. La division en grappes du groupe de travail permet de distribuer la charge des tâches de coordination aux nœuds élus comme des têtes de grappe. Comme tous les nœuds de la GRID disposent du service de coordination, elles peuvent toutes être élues comme tête de grappes selon leurs positions stratégiques dans les environnements des réseaux d'ordinateurs exploités par notre GRID. Rappelons que chaque tête de grappe est responsable de la coordination des nœuds de sa grappe et se charge de la redistribution des tâches qu'elle reçoit de l'ensemble de ses nœuds. Dans cette nouvelle génération, la grappe se partage le nombre total d'instructions informatiques reçues par sa tête de grappe proportionnellement à leurs indices d'élection. Pour cela, nous utilisons la relation (4.3) :

$$Taille'_{SousTache_n} = \frac{P'(n) \times N_{opérations}}{\sum_{x=1}^{v} P'(n_x)} \tag{4.3}$$

Où :

- $Taille'_{SousTâche_n}$ représente le nombre d'instructions informatiques formant la sous-tâche à traiter par le nœud *n*.
- $P'(n)$ représente la fonction permettant de déterminer la performance du nœud *n* dans sa grappe :

$$P'(n) = \frac{(G(n) + P(n))^{C_P}}{Q(n,t)^{C_Q} \times M(n)^{C_M}} \tag{4.4}$$

Avec :

- *G(n)* représentant la somme des performances *P'* des nœuds de sa grappe si *n* est lui-même une tête de grappe ou sinon *T(n)* est nul.

- C_P, C_M et C_Q représentant les coefficients attribués respectivement à l'importance de la performance, de la mobilité et de la qualité de service des connexions des nœuds.

- $N_{opérations}$ représente le nombre total d'instructions informatiques nécessaires pour traiter la tâche reçue par la tête de grappe.
- $\sum_{x=1}^{v} E'(n_x)$ représente la somme de la performance des *v* nœuds de la grappe.

La hiérarchisation des têtes de grappes permet de faciliter la division des tâches issues de la requête à traiter. La tête de grappe principale divise la tâche de la requête reçue en sous-tâches qui seront redistribués aux nœuds de sa grappe; si sa grappe compte des têtes de grappe parmi ses nœuds, à leur tour, elles diviseront la tâche qu'elles ont reçue en sous-tâches dans leurs propres grappes et ainsi de suite tant que tous les nœuds du groupe de travail n'ont pas reçu leurs tâches à réaliser.

Tout comme les GRID de troisième génération, le groupe de travail permet l'ajout dynamique de nœuds. Dans ce cas, il sera rattaché à une grappe dont la tête est la plus proche. En plus, les ajouts du support de la mobilité des nœuds et du travail « hors groupe » dans l'architecture de notre GRID permettent aux nœuds de ne pas arrêter le traitement de leurs tâches même s'ils se déconnectent de leurs têtes de grappe. À leur retour, s'il n'est pas trop tard, ils restitueront les résultats de leurs traitements (voir la section 3.2.1 pour plus de détails). De plus, les ajouts du support de la mobilité des tâches et du transfert de l'avancement des traitements dans notre architecture GRID permettent aux nœuds sachant qu'ils se déconnecteront bientôt de faire profiter la grappe de leurs réalisations avant de la quitter.

Comme nous l'avons fait pour les générations de GRID précédentes, nous considérons aussi les têtes de grappe dans la division du nombre total d'instructions informatiques de la requête. Si une tête de grappe offre le service de partage de sa puissance de calcul, il se considère automatiquement comme nœud de la grappe en plus de poursuivre son rôle de tête de grappe.

Pour clarifier les différentes étapes de ce mécanisme de traitement des tâches, les Tableaux de 4.16 à 4.20 spécifient le pseudo-code de la méthode *traiter_tâche* selon les services offerts par les nœuds.

Tableau 4.16 - Pseudo-code de la méthode *traiter_tâche* dans une GRID 4G (1re partie)

Étape	Service	Pseudo-code
Toutes	**Description :** Pendant toutes les étapes du traitement des tâches, les nœuds s'échangent des messages de signalisation afin de maintenir à jours leurs tables de routage et de réévaluer périodiquement la qualité de services des liens.	
	Pour tous les nœuds	▪ En parallèle : • Poursuivre l'échange de messages de signalisation décrite dans l'étape *2* du Tableau 4.8. • Évaluer à chaque échange de messages de signalisation la qualité de services des interconnexions.
1	**Description :** La tête de grappe principale reçoit une requête à traiter ou une tête de grappe reçoit une tâche d'une tête de grappe hiérarchiquement plus élevée. Elle divise le nombre total d'instructions informatiques à traiter en autant de sous-tâches qu'il y a de nœuds dans sa grappe.	
	Service de coordination	▪ Déterminer à l'aide de la relation (4.3), le nombre d'instructions informatiques de sa sous-tâche à traiter. ▪ Pour *chaque nœud de sa grappe* : • Déterminer à l'aide de la relation (4.3), le nombre d'instructions informatiques de la sous-tâche à traiter par ce nœud. • Envoyer au nœud sa sous-tâche à traiter.
	Partage de la puissance de calcul	▪ Attendre le nombre d'instructions informatiques à traiter dans la sous-tâche qui lui est assignée par sa tête de grappe.
2	**Description :** Les nœuds traitent leurs sous-tâches. Si un nœud ne répond plus aux messages de signalisation, la tête de grappe redistribue sa tâche à toute la grappe. S'il y a une nouvelle ressource informatique voisine d'une tête de grappe et que la performance de la GRID est inférieure à la valeur de la qualité de service de la requête, la tête de grappe ajoute le nouveau nœud à sa grappe. Quand toutes les instructions informatiques sont réalisées, les nœuds signalent la fin du traitement de leurs sous-tâches à leurs têtes de grappe.	
	Service de coordination	▪ En parallèle : • Tant *que la tête de grappe n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche* : ○ À *chaque seconde* : ♦ *… (Suite au Tableau 4.17)*

Tableau 4.17 - Pseudo-code de la méthode *traiter_tâche* dans une GRID 4G (2e partie)

Étape	Service	Pseudo-code
2	(Suite) Service de coordination	♦ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. • Si *un nœud ne répond pas à un message de signalisation* : o Le nœud est retiré de la liste de la grappe. o Pour *tous les nœuds restants de la grappe* : ♦ Déterminer à l'aide de la relation (4.3), le nombre d'instructions informatiques de la tâche du nœud déconnecté qui sont à redistribuer à ce nœud. ♦ Envoyer sa nouvelle sous-tâche à traiter. • Si *l'hôte a un nouveau voisin* : o Si *la qualité de service de la requête n'est pas définie*. Ou o Si *elle est définie et est supérieure à la performance du groupe de travail.* Alors ♦ Ajouter le nouveau nœud à sa grappe. ♦ Demander aux nœuds de la grappe, le nombre d'instructions informatiques qu'ils ont réalisées. ♦ Évaluer le nombre total d'instructions informatiques restant à traiter par la grappe. ♦ Pour *tous les nœuds de la grappe* : ◊ Envoyer un message d'arrêt du traitement en cours. ◊ Déterminer à l'aide de la relation (4.3), le nombre d'instructions informatiques restant à faire traiter par ce nœud. ◊ Envoyer la nouvelle sous-tâche à traiter au nœud. • Tant *que tous les nœuds n'ont pas signalé la fin de leurs traitements* : o Écouter les messages provenant des nœuds. ▪ Quand *tous les nœuds de la grappe ont fini leurs tâches* : • *… (Suite au Tableau 4.18)*

Tableau 4.18 - Pseudo-code de la méthode *traiter_tâche* dans une GRID 4G (3e partie)

Étape	Service	Pseudo-code
2	(Suite) Service de coordination	• Si *cette tête de grappe est la tête de grappe principale* : o Arrêter la simulation. • Sinon : o Signaler la fin du traitement de la tâche à la tête de grappe l'ayant envoyée. o Attendre une autre sous-tâche.
	Partage de la puissance de calcul	▪ Tant *que le nœud est dans une grappe*, en parallèle : • Si *le nœud reçoit une sous-tâche* : o Tant *que le nœud n'a pas fini de traiter toutes les instructions informatiques de sa sous-tâche* : ♦ À *chaque seconde* : ◊ Le nombre d'instructions informatiques traité augmente de 10^9 par 1 GHz de puissance de calcul. o Envoyer un message à l'hôte pour signaler la fin du traitement. • Si *le nœud reçoit un message d'arrêt de traitement* : o Arrêter le traitement de la tâche en cours. o Effacer toutes les tâches qui sont à traiter. • Si *le nœud se déconnecte de sa grappe* : o Poursuit le traitement des tâches en cours.
3	**Description :** Après une déconnexion, si le nœud ne se reconnecte pas trop tard au groupe de travail, il poursuit son traitement en cours.	
	Service de coordination	▪ Si *un nœud se reconnecte au groupe de travail après une déconnexion* : • Si *le nœud se reconnecte à son ancienne tête de grappe* : o Rajouter le nœud à sa grappe. o Demander aux nœuds de la grappe le nombre d'instructions informatiques qu'ils ont réalisées. o … *(Suite au Tableau 4.19)*

Tableau 4.19 - Pseudo-code de la méthode *traiter_tâche* dans une GRID 4G (4e partie)

<table>
<tr><th>Étape</th><th>Service</th><th>Pseudo-code</th></tr>
<tr><td rowspan="2">3</td><td>(Suite)
Service de coordination</td><td>

Évaluer le nombre total d’instructions informatiques restant à traiter par la grappe et le réduire du nombre d’instructions déjà traitées par le nœud qui s’est reconnecté.
Pour tous les nœuds de la grappe :

Envoyer un message d’arrêt du traitement en cours.
Déterminer à l’aide de la relation (4.3), le nombre d’instructions informatiques restant à faire traiter par ce nœud.
Envoyer la nouvelle sous-tâche à traiter.

• Si le nœud ne se reconnecte pas à son ancienne tête de grappe :

Demander le transfert de l’avancement de la réalisation de ses tâches à la bonne tête de grappe.
Considérer le nœud comme un nouveau nœud de sa grappe.

</td></tr>
<tr><td>Partage de la puissance de calcul</td><td>
▪ Si le nœud se reconnecte au groupe de travail après une déconnexion :

Poursuivre le traitement des tâches en cours.
Attendre les messages de synchronisation de la tête de grappe.

</td></tr>
<tr><td rowspan="2">4</td><td colspan="2">Description : Un nœud ayant décidé de se déconnecté procède au transfert de l’avancement de ses tâches.</td></tr>
<tr><td>Service de coordination</td><td>
▪ Si la tête de grappe reçoit un transfert de l’avancement d’une tâche :

Retirer le nœud de la grappe si ce n’est pas déjà fait.
Demander aux nœuds de la grappe le nombre d’instructions informatiques qu’ils ont réalisées.
Évaluer le nombre total d’instructions informatiques restant à traiter par la grappe et le réduire du nombre d’instructions déjà traitées dans l’avancement des tâches du nœud qui se déconnecte bientôt.
Pour tous les nœuds de la grappe :

Envoyer un message d’arrêt du traitement en cours.
Déterminer à l’aide de la relation (4.3), le nombre d’instructions informatiques restant à faire traiter par ce nœud.
Envoyer la nouvelle sous-tâche à traiter au nœud.

</td></tr>
</table>

Tableau 4.20 - Pseudo-code de la méthode *traiter_tâche* dans une GRID 4G (5e partie)

Étape	Service	Pseudo-code
4	Partage de la puissance de calcul	▪ Si *la couche « ressource » de l'architecture de notre GRID met fin au service de partage de la puissance de calcul* : • Arrêter et transférer l'avancement du traitement de ses tâches à sa tête de grappe. • Effacer toutes les tâches qui sont à traiter.

4.1.7 Interfaces graphiques et validation

Même si nous n'avons pas présenté le pseudo-code de toutes les autres classes de l'architecture de notre simulateur de GRID, nous avons dû les implémenter afin de rendre fonctionnel notre outil de simulation.

Comme processus de validation simple, nous allons reprendre la liste des spécifications que nous avons formulée dans la section 4.1.1 afin de nous assurer que toutes les fonctionnalités attendues soient présentes dans les interfaces graphiques du simulateur. Tout d'abord, nous commencerons par présenter l'interface principale du simulateur. Puis pour chaque élément de celui-ci, nous situerons les fonctionnalisées qu'ils offrent.

4.1.7.1 Interface principale du simulateur

L'interface principale du simulateur est composée de 5 « vues ». Nous entendons par une « vue », un regroupement fonctionnel et simple à utiliser d'outils ou d'informations. Ces regroupements permettent une meilleure organisation de l'espace et des fonctionnalités offertes, de plus ils rendent plus agréable l'expérience d'interaction de l'utilisateur. La Figure 4.4 représente l'interface principale de notre simulateur de GRID et y indique les regroupements suivants :

1. Projets : Représentation sous forme d'arbre des projets en cours. Pour chaque projet de simulation, l'utilisateur peut naviguer entre ses différents réseaux d'ordinateurs à l'aide de sa souris.
2. Réseau d'ordinateurs : Représentation visuelle du réseau d'ordinateurs en cours de simulation. À l'aide d'onglet, l'utilisateur peut naviguer entre les simulations de réseaux d'ordinateurs qu'il a ouverts.

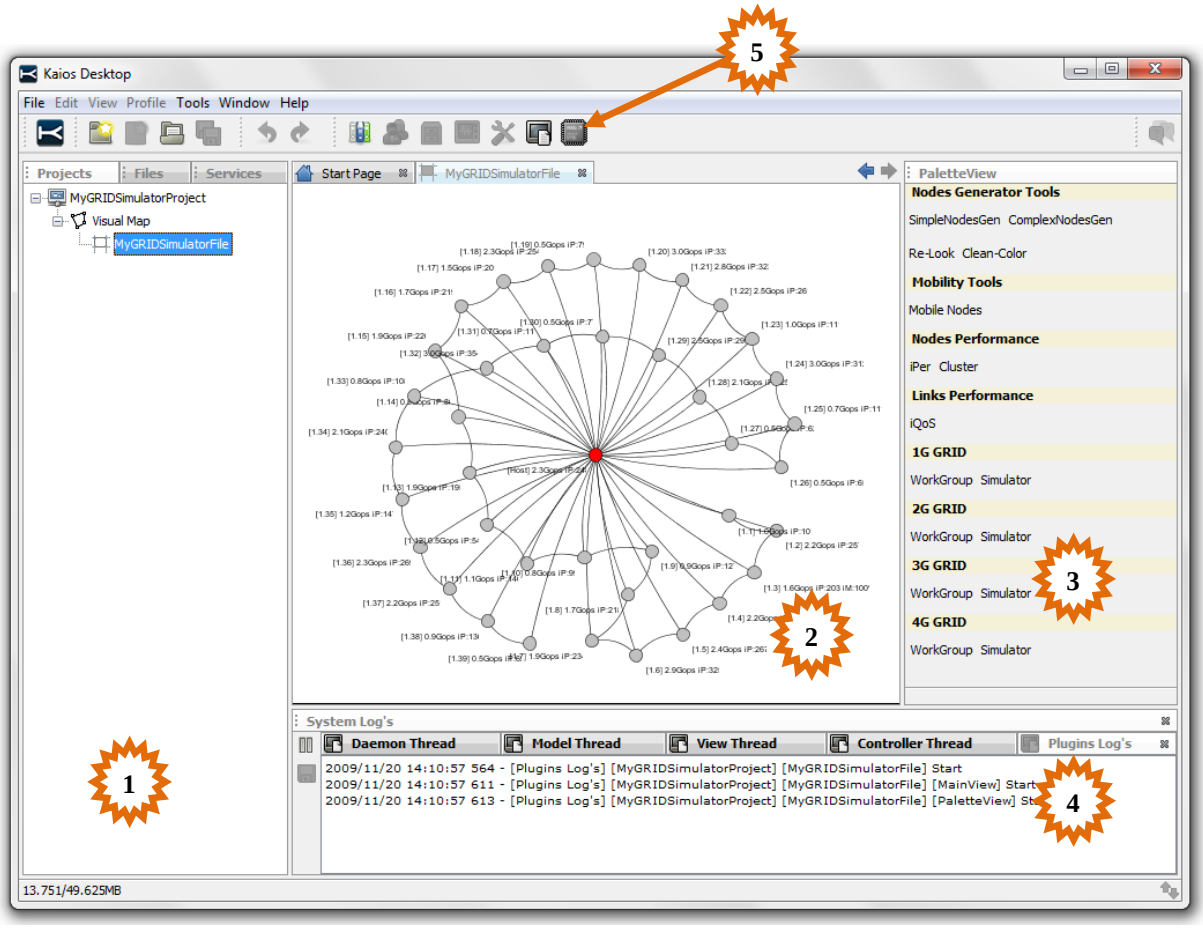


Figure 4.4 - Interface principale du simulateur de GRID

3. Palette d'outils de simulation : Liste des outils de simulation organisés selon leurs fonctionnalités. Nous y retrouvons les outils pour construire et manipuler un réseau de simulation, pour définir la mobilité des ressources, les caractéristiques des ressources, les caractéristiques des liens et pour choisir les étapes de simulation selon la génération de GRID que nous voulons simuler dans notre réseau d'ordinateurs.
4. Log et résultats : Représentation textuelle des logs d'utilisation des outils de l'interface et présentation des résultats de simulations.
5. Barre d'outils générales : Liste des outils permettant entre autres d'ouvrir un projet, de créer un projet, de créer un réseau d'ordinateurs, de sauvegarder un réseau d'ordinateurs et de quitter le simulateur.

Nous proposons dans l'Annexe A, plusieurs saisies des interfaces du simulateur de GRID que nous avons développé pour ce mémoire.

4.1.7.2 Validation du simulateur

Pour chaque regroupement de l'interface principale présenté précédemment, nous énumérons dans l'Annexe B, les spécifications des fonctionnalités qu'il offre afin de valider le fonctionnement global de notre simulateur de GRID.

Toutes les fonctionnalités attendues étant présentes dans notre outil de simulation de GRID, nous pouvons commencer l'évaluation et l'analyse des performances des différentes générations de GRID que nous avons implémentées.

4.2 Évaluation et analyse des performances

Notre outil de simulation de GRID étant fonctionnel, dans la deuxième partie de ce chapitre, nous présentons l'évaluation des quatre générations de GRID que nous avons implémentées. Une simulation de GRID débute par la formation d'un réseau d'ordinateurs. Après avoir identifié le nœud qui recevra la requête du client, nous choisissons la génération de GRID que nous désirons simuler et nous lançons la simulation. Durant la simulation, des messages seront échangés, des tâches distribuées et des instructions traitées. Mis à part l'observation visuelle de ces activités, l'objectif principal de la simulation est de quantifier les performances de ces activités en évaluant les retards engendrés par les échanges de messages, les délais de transfert des sous-tâches aux ressources informatiques et le temps qui leur est nécessaire pour les traiter.

Le délai de transfert d'un message ou d'une tâche dans une interconnexion entre deux nœuds est calculé par les méthodes *envoyer_message_A_B* (pour envoyer un message ou une tâche du nœud A à B) et *envoyer_message_B_A* (pour envoyer un message ou une tâche du nœud B à A) de la classe « Lien ». Pour ce faire, nous utilisons la relation (4.5) :

$$Délai_{Transfert} = Temps_{Réponse} + Taille_{Message} \times Débit \qquad (4.5)$$

Avec :

- $Délai_{Transfert}$ représentant le délai de temps nécessaire pour que le destinataire reçoive le message ou la tâche ;
- $Délai_{Réponse}$ représentant le délai de réponse du lien. Il est de valeur différente selon le sens de la connexion ;
- $Taille_{Message}$ représentant la taille du message à envoyer. Pour les messages autres que des tâches, nous avons spécifié dans la supposition A.IX de la section 4.1.2, qu'ils

sont de tailles négligeables donc égales à zéro. Pour les tâches, la taille d'un message correspond aux nombres d'instructions de la tâche ;

- *Débit* représentant la vitesse de transfert du lien. Elle est de valeur différente selon le sens de la connexion.

Dans l'implémentation des classes des intergiciels de GRID, nous avons défini les principaux mécanismes qui permettent la formation d'un groupe de travail puis le traitement d'une requête par celui-ci. Nous exposons dans une première phase, l'évaluation et l'analyse des performances des GRID à former leurs groupes de travail. Puis en seconde phase, nous évaluons et analysons les performances des différentes générations dans le traitement d'une requête. Dans ces deux phases, nous nous concentrons sur l'évolution des performances selon la génération des GRID, la taille de la GRID, la taille des requêtes et le type d'environnement.

Tableau 4.21 - Configuration du réseau d'ordinateurs

Paramètres	Valeurs
Indice de performance	➢ $I_{Performance_{CPU_n}}$ = 1 GHz, $I_{Performance_{RAM_n}}$ = 1 Go, $I_{Performance_{HD_n}}$ = 10 Go et $I_{Performance_{E_n}}$ = 100 %. ➢ *P(n)* = 130 avec α_1, α_2, α_3 et α_4 égaux à 100, 10, 1 et 10.
Indice de qualité de service des liens	Pour tous les voisins *n'* de *n* : ➢ $I_{QdS_{Débit_{nn'}}}$ = 1 Mo/s et $I_{QdS_{Délai_{nn'}}}$ = 10 ms. ➢ *Q(n,n')* = 20 *avec* $\beta_1$et β_2 égal à 10 et 1.
Indice d'élection	➢ C_P = 1, C_Q = 0.05 et C_M = 0.1.

4.2.1 Formation des groupes de travail

Pour évaluer la performance de formation d'un groupe de travail, nous mesurons la période de temps nécessaire à la GRID pour former un groupe de travail dès qu'elle reçoit une requête. Pour ce faire, nous lançons un chronomètre dans la méthode *former_groupe* des classes d'intergiciels de GRID du serveur de coordination ou de l'hôte. Nous ne mesurons pas les étapes de formation de la GRID, le compteur débute dès qu'une requête sera reçue et s'arrêtera à la fin de la méthode. Pour chaque évaluation, nous présentons la moyenne de 10 simulations et nous utilisons les paramètres du Tableau 4.21 pour toutes les ressources *n* du réseau d'ordinateurs.

Nous débutons la première évaluation de performance des GRID en suivant l'évolution du délai de temps qui leur est nécessaire pour former un groupe de travail selon la taille de la GRID. Pour cette évaluation, nous considérons la configuration du réseau d'ordinateurs présenté dans le Tableau 4.21. De plus : toutes les ressources sont voisines, nous ne spécifions pas la qualité de service de la requête, la taille de la requête est de $50x10^9$ instructions informatiques et la GRID est dans un environnement stable (d'où pour toutes les ressources *n*, *M(n)* = 0 %). Nous obtenons les résultats présentés à la Figure 4.5.

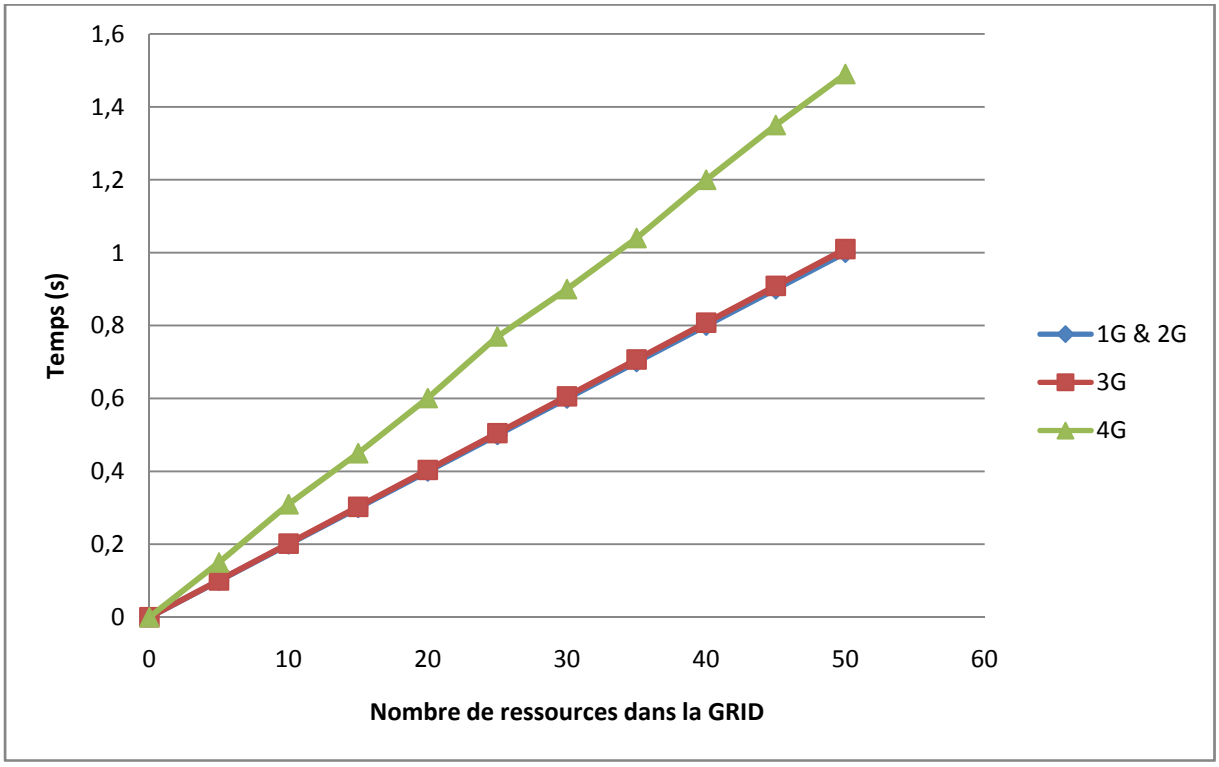


Figure 4.5 - Performances de toutes les générations de GRID dans la formation d'un groupe de travail selon leurs tailles

Dans ce graphique, les courbes de performance des trois premières générations sont identiques. Cela s'explique par la similitude de leurs mécanismes de formation de groupe de travail. La nouvelle génération de GRID prend un retard dans la formation d'un groupe de travail allant jusqu'à 50 % par rapport aux performances des GRID précédentes. Ce retard est engendré par le calcul des indices d'élection.

Notre génération de GRID est la seule à pouvoir exploiter des ressources qui ne sont pas voisines de l'hôte. Comme deuxième évaluation, nous suivons l'évolution de la performance de notre génération de GRID dans la formation d'un groupe de travail selon le nombre de têtes de

grappe dans celui-ci. Nous utilisons la configuration précédente du réseau d'ordinateurs avec 30 ressources dans un environnement stable (d'où pour toutes les ressources *n*, *M(n)* = 0 %). De plus, nous essayons de former des grappes de même taille, par exemple s'il y a deux têtes de grappe, il y aura 15 nœuds dans chaque grappe. Nous obtenons les résultats présentés à la Figure 4.6.

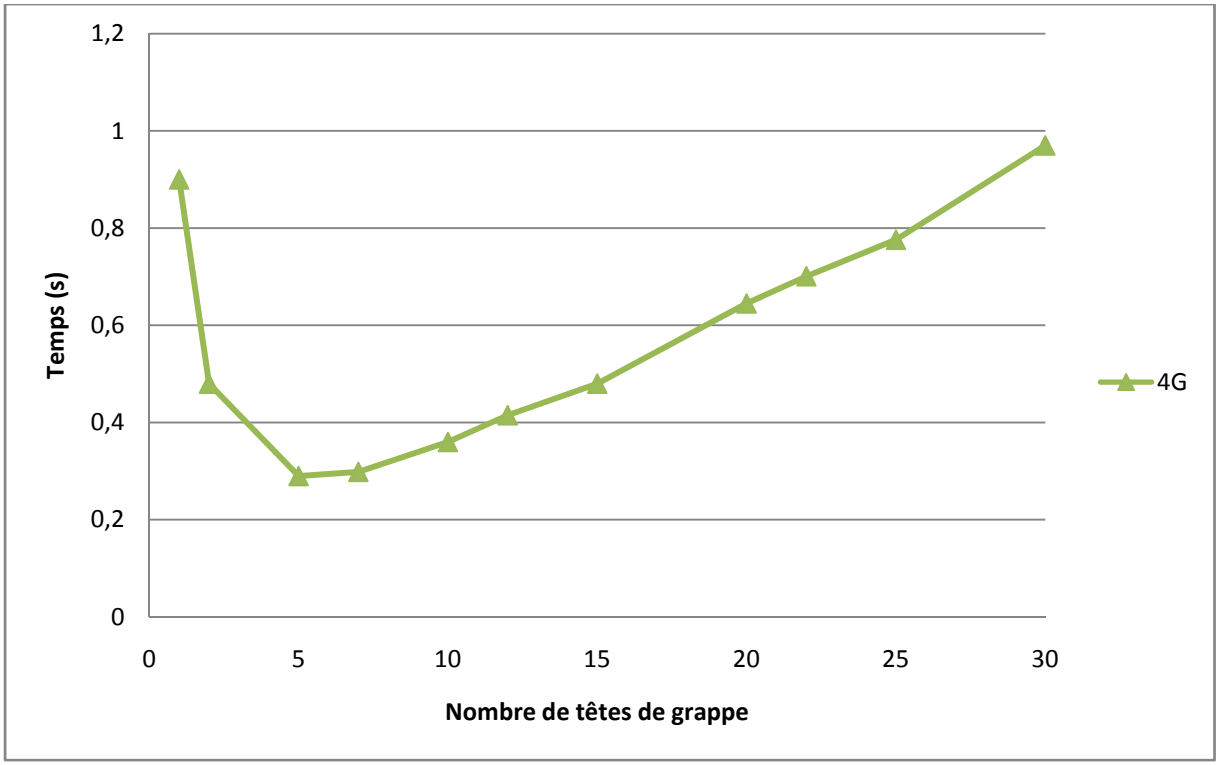


Figure 4.6 - Performance de la nouvelle génération de GRID dans la formation d'un groupe de travail selon le nombre de têtes de grappe.

Nous remarquons que la performance de notre architecture GRID à former un groupe de travail s'améliore avec l'ajout de quelques têtes de grappe. En effet, avec 5 têtes de grappe (dont l'hôte) dans la GRID, la nouvelle génération de GRID rassemble 30 nœuds aussi vite qu'environ 15 nœuds dans les groupes de travail des générations de GRID précédentes. Cependant, les améliorations de performance de la nouvelle génération de GRID se réduisent au point de s'annuler si le nombre de têtes de grappe augmente trop par rapport au nombre de nœuds dans la GRID. Selon nos mesures actuelles, le ratio le plus efficace est d'une tête de grappe pour 6 nœuds de la GRID. Si nous changeons de réseau d'ordinateur, ce ratio changera aussi.

Afin de déterminer la variation des performances des GRID selon la taille d'une requête et le type d'environnement, nous passons à la phase d'évaluation et d'analyse suivante.

4.2.2 Traitements des requêtes

Pour évaluer la performance du traitement d'une requête, nous mesurons la période de temps nécessaire pour réaliser le mécanisme de la méthode *traiter_tâche* des classes d'intergiciels de GRID du serveur de coordination ou de l'hôte. Pour chaque évaluation, nous présentons la moyenne de 10 simulations, nous ne spécifions pas la qualité de service de la requête et nous utilisons la configuration du réseau d'ordinateurs présenté au Tableau 4.21.

La troisième évaluation de performance portera sur l'évolution de la performance des GRID selon la taille d'une requête et le type d'environnement. Pour ce faire, selon le type d'environnement, nous procédons à l'évaluation de la performance des GRID à traiter des requêtes de différentes tailles. Le Tableau 4.22 présente un bilan des paramètres complétant la configuration des 30 ressources *n* du réseau d'ordinateur que nous utiliserons dans nos simulations.

Tableau 4.22 - Indice de mobilité des ressources selon le type d'environnement

Paramètres	Valeurs
Indice de mobilité	➢ *Environnement stable* : $M(n)$ = 0 %, $P_{Déconnexion\_Douce_n}$ = 100 % (voir équation 4.1) et la probabilité d'une reconnexion du nœud (Pr) de 0 %. ➢ *Environnement faiblement dynamique* : $M(n)$ = 20 %, $P_{Déconnexion\_Douce_n}$ = 80% et $Pr(n)$ = 20 %. ➢ *Environnement dynamique* : $M(n)$ = 50 %, $P_{Déconnexion\_Douce_n}$ = 50% et $Pr(n)$ = 50 %. ➢ *Environnement mobile* : $M(n)$ = 80 %, $P_{Déconnexion\_Douce_n}$ = 20% et $Pr(n)$ = 80 %.

À cela, nous ajoutons que le serveur de coordination et l'hôte ne sont pas mobiles. Ils ont donc un indice mobilité nul afin d'éviter l'arrêt prématuré des simulations en cours. Nous obtenons les résultats présentés aux Figures 4.7 à 4.10.

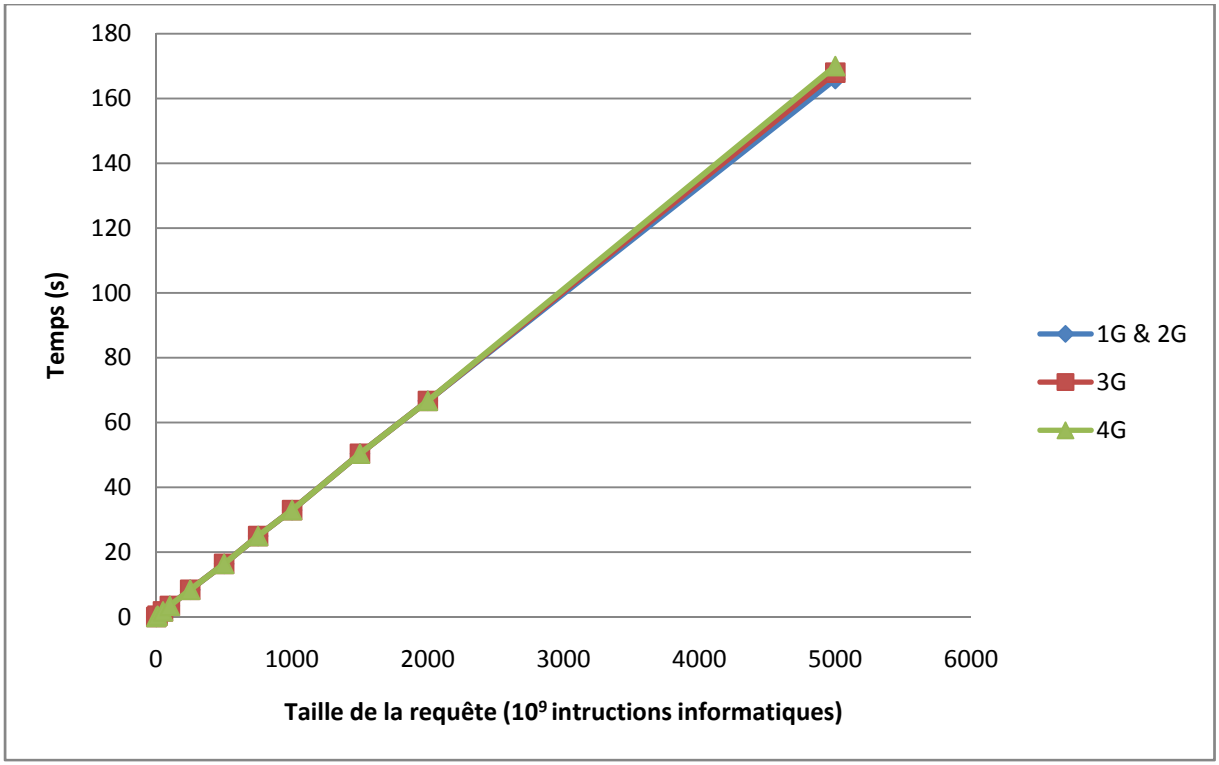


Figure 4.7 - Performance des GRID dans un environnement stable

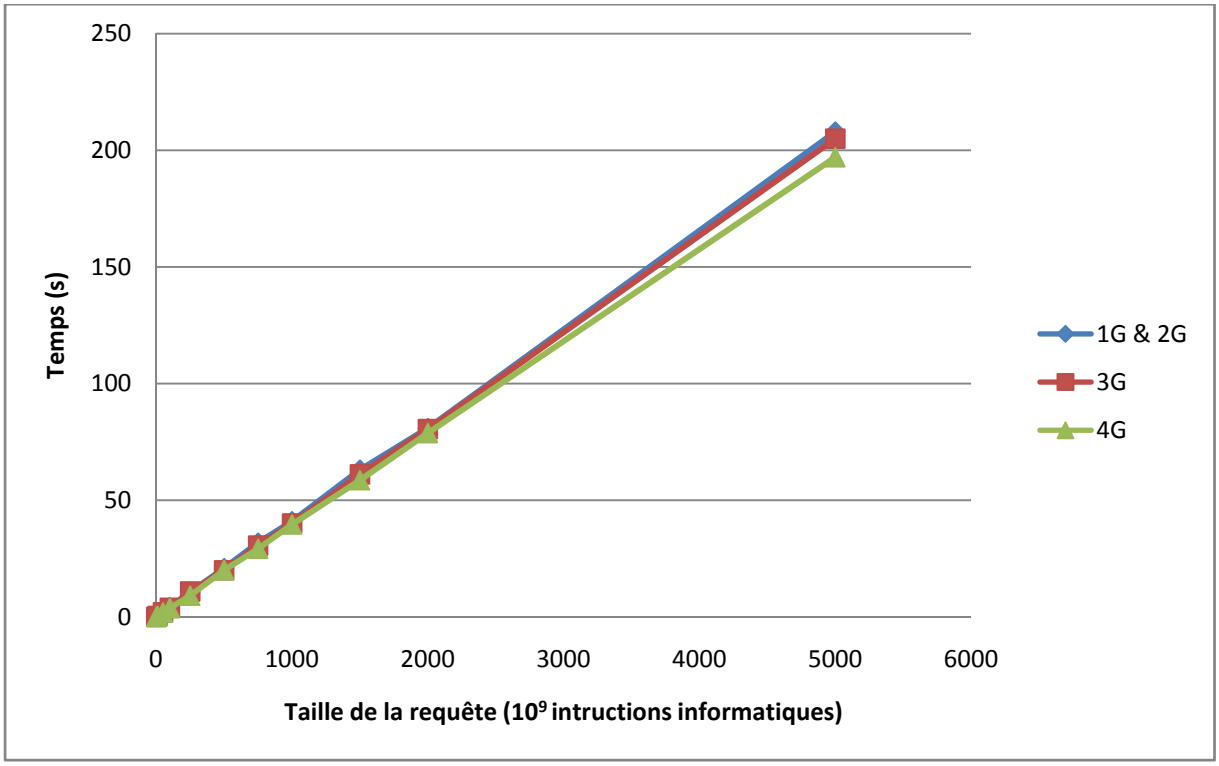


Figure 4.8 - Performance des GRID dans un environnement faiblement dynamique

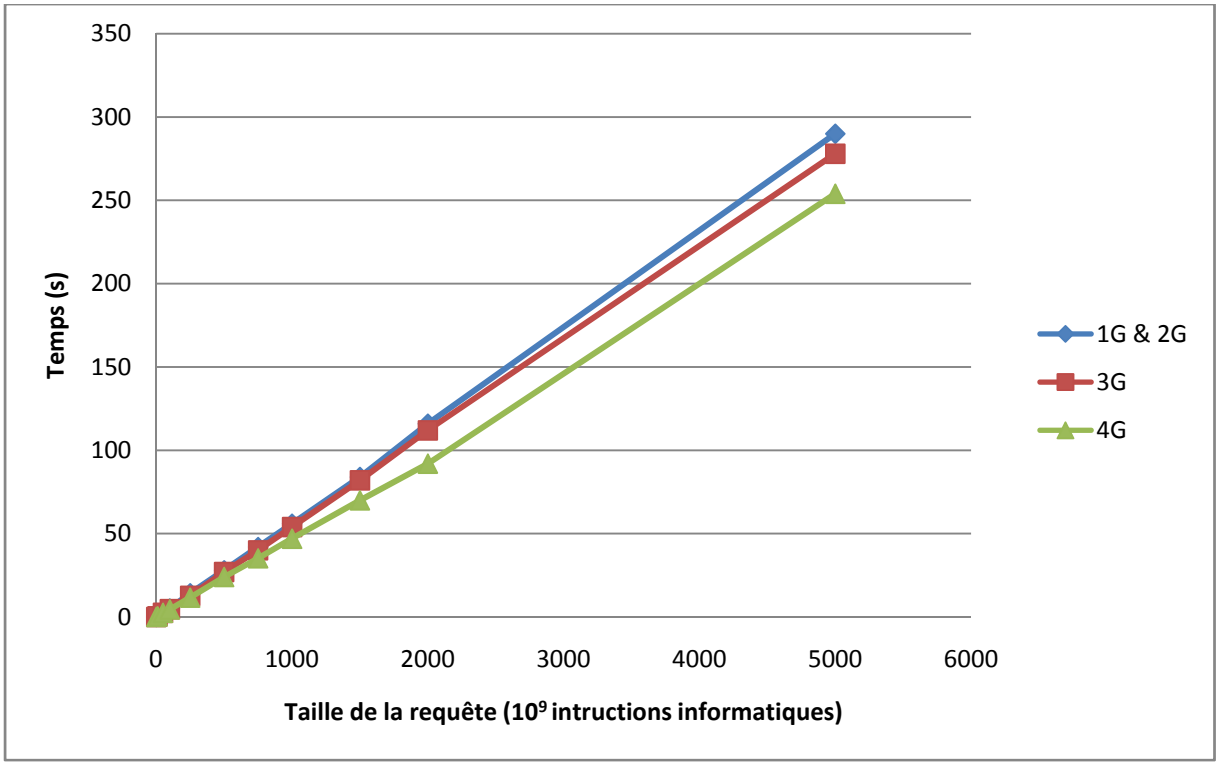


Figure 4.9 - Performance des GRID dans un environnement dynamique

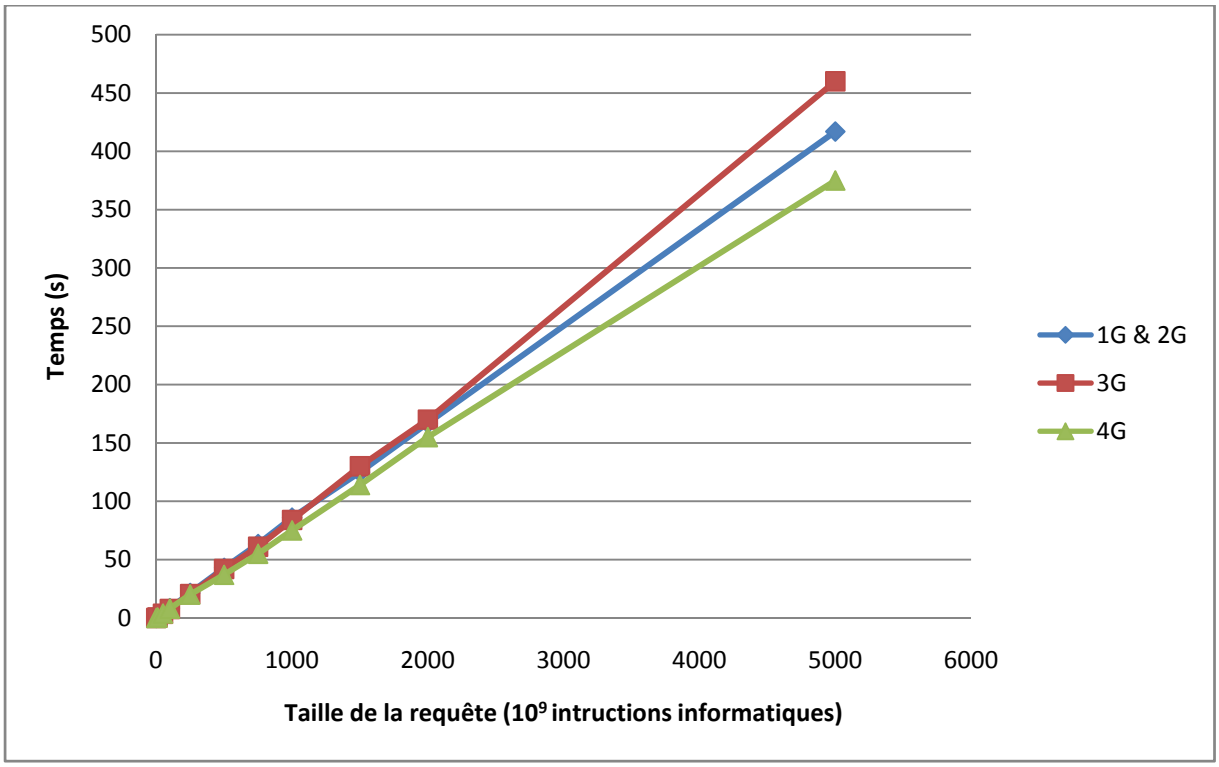


Figure 4.10 - Performance des GRID dans un environnement mobile

Suite à l'analyse des 4 Figures précédentes, dans un environnement stable, les quatre générations de GRID ont des performances similaires. Dans des environnements dynamiques, l'écart se creuse entre les générations. En général, plus l'indice de mobilité des ressources augmente, plus le délai de traitement d'une requête augmente aussi. Dans nos mesures, notre architecture GRID est toujours la plus performante. Dans le meilleur des cas, nous notons une amélioration de 18 % dans un environnement mobile par rapport à la troisième génération et de 13 % dans un environnement dynamique par rapport aux GRID de première et deuxième générations.

Comme dernière évaluation, nous mesurons l'évolution des performances d'une GRID de nouvelle génération à traiter des requêtes de différentes tailles selon le type d'environnement. Durant nos simulations, la GRID est formée de 30 nœuds regroupés en 5 grappes de 6 nœuds et nous utilisons les paramètres des Tableaux 4.21 et 4.22. Nous obtenons les résultats présentés à la Figure 4.11, un agrandissement du graphique obtenu est disponible en Annexe C.

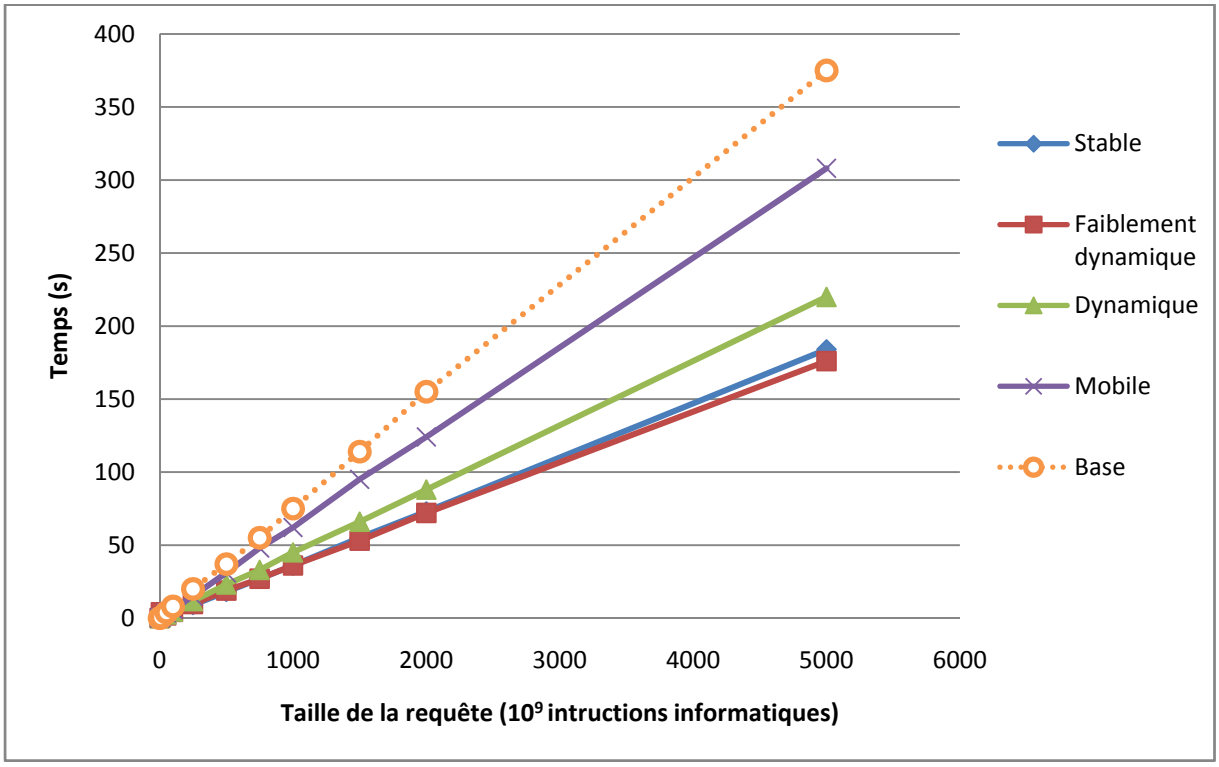


Figure 4.11 - Performance de la nouvelle génération de GRID selon l'environnement

La courbe « Base » représente le groupe de travail de la nouvelle génération de GRID dans un environnement mobile mesuré dans l'évaluation précédente (voir Figure 4.10). Nous

remarquons que si nous ajoutons quelques têtes de grappes dans le groupe de travail, nous améliorons encore la performance de notre GRID dans des environnements dynamiques. Nous notons un gain de 22 % par rapport aux mesures présentes de la même génération et une amélioration de 48 % par rapport aux GRID de troisième génération.

Nos évaluations nous ont permis de vérifier les améliorations apportées à notre architecture de GRID. Nous avons été surpris de remarquer que les GRID de troisième génération sont moins performantes dans des environnements fortement dynamiques et mobiles que les GRID de première et deuxième génération. Même si la troisième génération de GRID permet l'ajout dynamique de ressources informatiques dans le groupe de travail, elle perd beaucoup de temps à redistribuer des tâches à ces nouveaux nœuds qui ne resteront connectés que peu de temps. Cependant même si une partie de notre architecture GRID est héritée de cette génération, les améliorations de performance observées nous laissent croire que nous avons atteint nos objectifs. Dressons le bilan de notre mémoire au chapitre suivant.

CHAPITRE 5
CONCLUSION

Dans ce dernier chapitre de notre mémoire, nous allons faire un bref rappel des objectifs que nous nous sommes fixés, rappeler les particularités de l'architecture GRID que nous avons proposées, discuter des limitations de nos travaux ainsi que des résultats que nous avons obtenus. Et pour terminer, nous compléterons notre recherche en suggérant des améliorations futures de notre architecture GRID.

5.1 Synthèse des travaux

Au début de notre recherche, nous avons établi que les GRID existantes ne nous permettaient pas d'exploiter les ressources mobiles efficacement. L'émergence des micro-ordinateurs portables et des technologies de communication sans fil contribuait aux limitations de l'usage des GRID. Malgré les améliorations des GRID de troisième génération apportées au domaine des réseaux d'ordinateurs, nous avons identifié certaines lacunes dans les mécanismes de gestion des sous-tâches. Dans un environnement où il y a fréquemment des déconnexions entre les liens des ressources de la GRID, le retard dans le traitement d'une requête augmente significativement. Nous avons aussi identifié que la mobilité des ressources n'était pas prise en considération. Il n'y a aucun mécanisme pour essayer de prévenir la fin de partage des ressources d'un nœud afin de limiter son utilisation par la GRID. La problématique de la faible performance des GRID de troisième génération dans des environnements dynamiques et mobiles nous a encouragé à fixer des objectifs d'amélioration de l'architecture des GRID. Nous nous sommes proposé de réaliser une architecture GRID de nouvelle génération efficace dans des environnements dynamiques et mobiles, cela, afin de supporter de nouvelles applications mobiles qui sont de plus en plus populaires. Dans une première phase, nous avons amélioré le support de la mobilité des tâches et des ressources des GRID de troisième génération. Suite à cela, nous avons introduit des concepts de mobilité dans l'architecture des GRID existante. Nous avons aussi ajouté dans l'architecture d'un nœud : l'habilité de transférer l'avancement des tâches à d'autres nœuds, l'habilité de poursuivre le traitement des tâches déjà en cours sans nécessairement être connecté en permanence au groupe de travail, et, pour finir, l'habilité de coopérer directement avec d'autres nœuds du groupe de travail sans passer par l'hôte. L'architecture réseau a aussi été révisée afin de

marquer le changement de génération de GRID. Le groupe de travail a été hiérarchisé par un ensemble de têtes de grappe et chaque tête de grappe représente un regroupement de ressources. Le nombre de têtes de grappe dans un groupe de travail et de ressources formant une grappe n'est pas fixe, la topologie se base sur des indices de performances et de qualité de services pour s'adapter aux environnements des différentes ressources de la GRID.

5.2 Limitations des travaux

Afin de tester notre proposition, nous avons développé un simulateur dans lequel nous avons implémenté le comportement général des différentes générations, dont celle que nous proposons. Afin de simplifier la réalisation de ces simulations, nous n'avons pas considéré la coopération entre tâches, bien que nous ayons proposé des améliorations à ce sujet. Nous nous sommes aussi limités à deux services par nœud. Le premier service est celui permettant la découverte et la coopération des nœuds, autrement dit celui qui permet à un nœud de devenir une tête de grappe. Le deuxième service implémenté est celui du partage de la puissance de calcul. Comme il est le service le plus sensible aux performances des GRID. En revanche, malgré ses limitations, notre outil de simulation nous a permis de prédire le comportement des GRID avec une forte population de ressources et dans différents environnements.

L'évaluation de notre architecture GRID démontre que notre GRID de nouvelles générations se comporte aussi bien qu'une GRID de troisième génération dans un environnement stable, peu dynamique et sans ressources distantes. Il est en effet difficile de dépasser les performances des GRID déjà existantes dans des environnements locaux. La qualité de service de ces réseaux y étant fixe et élevée, les nouveaux concepts introduits dans notre architecture GRID n'ont pas d'impact sur le fonctionnement global de la GRID. La topologie sera centralisée autour d'une seule tête de grappe, l'hôte.

Les améliorations des performances de notre architecture GRID par rapport aux autres générations de GRID sont visibles dès que l'environnement devient plus dynamique et que nous considérons des ressources distantes. La qualité de service des liens entre des nœuds de différents réseaux est rarement idéale. Pour communiquer entre deux ressources, les messages ne passent plus par un ou deux câbles d'une même technologie, mais par un ensemble de chemins possibles de différente qualité de service et de technologie. Dans cette situation, la GRID se trouve à cheval entre deux environnements : le premier stable formé par les réseaux locaux et proches, et un

deuxième environnement moins stable formé par l'ensemble des réseaux intermédiaires entre les ressources locales et distantes. C'est grâce à la prise en considération de la qualité de service des liens, que les tailles des sous-tâches envoyées aux ressources distantes sont réduites afin de ne pas ralentir le traitement global de la requête. L'amélioration des performances est encore plus significative lorsque le nombre de têtes de grappe ou les indices de mobilité des nœuds augmentent. Par exemple, en considérant que 80 % des nœuds de la GRID sont dans un environnement dynamique, nous améliorons dans le meilleur des cas mesurés de 48 % en moyenne la performance des GRID existantes.

La topologie de notre GRID étant dynamique, le nombre de messages de signalisation échangés entre les nœuds est important. Nous avons noté qu'il y avait un ralentissement des performances de notre GRID au début des traitements d'une requête. Le délai pour initialiser un groupe de travail dans notre GRID est nettement plus élevé que le délai nécessaire pour un groupe de travail d'une GRID de génération précédente qui est quasi instantané. Nous avons observé des ralentissements des performances de notre GRID pouvant aller jusqu'à 50% comparativement à celle des GRID de génération précédente. Ce comportement de notre GRID qui semble désavantageux est en fait un faible coût initial à payer afin d'organiser toutes les ressources de la GRID convenablement. Une fois le groupe de travail formé, les performances qui avait été réduites de quelques secondes rattrapent rapidement celles des GRID des générations précédentes.

Dans les réseaux d'ordinateur réel, ce retard permet à la topologie du groupe de travail de notre GRID d'utiliser des ressources voisines à toutes les têtes de grappe du groupe du travail contrairement aux autres GRID qui ceux limite à ceux du serveur de coordination ou de l'hôte. La population de ressources de notre GRID est donc agrandie, et les performances pouvant être atteinte largement supérieures à celle des GRID des générations précédentes.

5.3 Indications de recherches futures

Idéalement, notre GRID dans un environnement dynamique et mobile devrait maintenir de manière transparente la même performance que dans un environnement stable. La calibration manuelle des coefficients permettant de calculer nos indices de performances et d'élections est fastidieuse. La performance de notre GRID varie fortement avec le choix des valeurs de ses coefficients. Une amélioration fortement envisageable serait d'automatiser la calibration des

coefficients selon le type d'application à traiter et la qualité de service des interconnexions afin d'améliorer le choix de la topologie pour chaque environnement.

Une seconde amélioration envisageable serait d'optimiser les échanges des messages de signalisations. Ils sont très volumineux et couteux dans nos évaluations. Il existe des méthodes dans la littérature qui permettent de réduire l'envoi de messages identiques à un ensemble de nœuds autre que la méthode actuellement utilisée, l'« Unicast ».

Le choix des têtes de grappe d'un groupe de travail se fait en choisissant les nœuds avec le meilleur indice d'élection dans l'espoir d'obtenir la meilleure topologie possible. En pratique, nous obtenons une topologie des ressources de la GRID satisfaisante, mais très souvent pas la meilleure. L'utilisation des métaheuristiques dans le choix de ces têtes de grappe pourrait encore améliorer les performances de notre architecture GRID. De plus, notre prise en considération de l'énergie résiduelle des nœuds permettra d'obtenir une topologie moins gourmande en énergie tout en étant plus efficace.

Finalement, une quatrième amélioration possible serait de simplifier l'utilisation des vieux standards de programmation parallèle afin de faciliter l'accès aux GRID. Il existe un très grand nombre de micro-ordinateurs dont les puissances de calcul restent inutilisées et dont leurs propriétaires n'attendent que de les rentabiliser !

BIBLIOGRAPHIE

[1] G. A. Blaauw and F.P. Brooks Jr. "The Structure of System/360, Part I-Outline of the Logical Structure", IBM Systems Journal, Vol. 3, No. 2, pp. 119–135, 1964.

[2] W. Scott. "Intel's Pentium 4 3.2 GHz processor", Intel Tech. Report, Juin 2003.

[3] D. E. Bodenstab, T. F. Houghton, K. A. Kelleman, G. Ronkin, et E. P. Schan (October 1984). "UNIX Operating System Porting Experiences", AT&T Bell Laboratories Tech. Report, Vol. 63, No. 8, Part 2, pp. 9, 2004.

[4] Parallel Vitual Machine : http://www.csm.ornl.gov/pvm/ (accédé en septembre 2009).

[5] N. P. Kronenberg, H. M. Levy et W. D. Strecker. "VAXcluster: A closely-coupled distributed system", ACM Transactions on Computer Systems, 1986.

[6] TCP/IP, IETF : http://www.ietf.org/rfc/rfc0675.txt (accédé en septembre 2009).

[7] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky et D.Werthimer. "SETI@home: An Experiment in Public- Resource Computing, Communications of the ACM", Vol. 45, No. 11, ACM Press, USA, 2002.

[8] R. Buyya. "GRIDBus: Convergence Characteristics for Clusters, Grids, and P2P networks, Grid Computing and Distributed Systems Lab", The University of Melbourne, Melbourne, Australie, 2002.

[9] A. Chien, B. Calder, S. Elbert et K. Bhatia. "Entropia: Architecture and Performance of an Enterprise Desktop Grid System, Journal of Parallel and Distributed Computing", Academic Press, Vol. 63, No. 5, USA, Mai 2003.

[10] C. Germain, V. Neri, G. Fedak et F. Cappello. "XtremWeb: Building an experimental platform for Global Computing", Proc. of the 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000), Bangalore, Inde, Décembre 2000.

[11] S. M. Larson, C. D. Snow, M. R. Shirts et V. S. Pande. "Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology", Computational Genomics, Richard Grant (ed.), Horizon Press, 2002.

[12] Distributed.net : http://www.distributed.net/ (accédé en décembre 2008).

[13] F. Cappello. "3rd Generation Desktop Grids", Proc. of 1st XtremWeb Users Group Workshop (XW'07). Hammamet, Tunisie, 2007.

[14] A. Luther, R. Buyya, R. Ranjan et S. Venugopal. "Alchemi: A .NET-Based Enterprise Grid Computing System", Proc. of the 6th International Conference on Internet Computing (ICOMP'05), Las Vegas, USA, 2005.

[15] D. P. Anderson. "BOINC: A System for Public-Resource computing and Storage", Proc. of 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh,USA, Novembre 2004.

[16] Intel Corporation, United Devices' Grid MP on Intel Architecture : http://www.ud.com/rescenter/files/wp_intel_ud.pdf (accédé en décembre 2008).

[17] M. Firuziaan, O. Nommensen. "Parallel Processing via MPI & OpenMP", Linux Enterprise, Octobre 2002.

[18] D.M. Jones. "The New C Standard: A Cultural and Economic Commentary", Addison-Wesley, Vol. 32, No. 5, pp. 961-975, 2009.

[19] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Nri et O. Lodygensky. "Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid", Journal of Future Generation Computer Systems, Elsevier, Vol. 21, No. 3, pp. 417-437, Mars 2005.

[20] X. Ma, V. W. Freeh, T. Yang, S. S. Vazhkudai, T. A. Simon et S. L. Scott. "Coupling prefix caching and collective downloads for remote dataset access", Proceedings of the 20th annual international conference on Supercomputing(ICS'06), pp 229-238, Queensland, Australie, 2006.

[21] A. Rezmerita, V. Neri, F. Cappello. "Toward Third Generation Internet Desktop Grids", Rinria Tech. Report, Mai 2007.

[22] X. Chu, K.a Nadiminti, C. Jin et al. "Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications", Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, pp. 151-159, 2007.

[23] G. Fedak, H. He, F. Cappello. "BitDew: A data management and distribution service with multi-protocol file transfer and metadata abstraction", J. Network and Computer Applications, Vol. 32, No. 5, pp. 961-975, 2009.

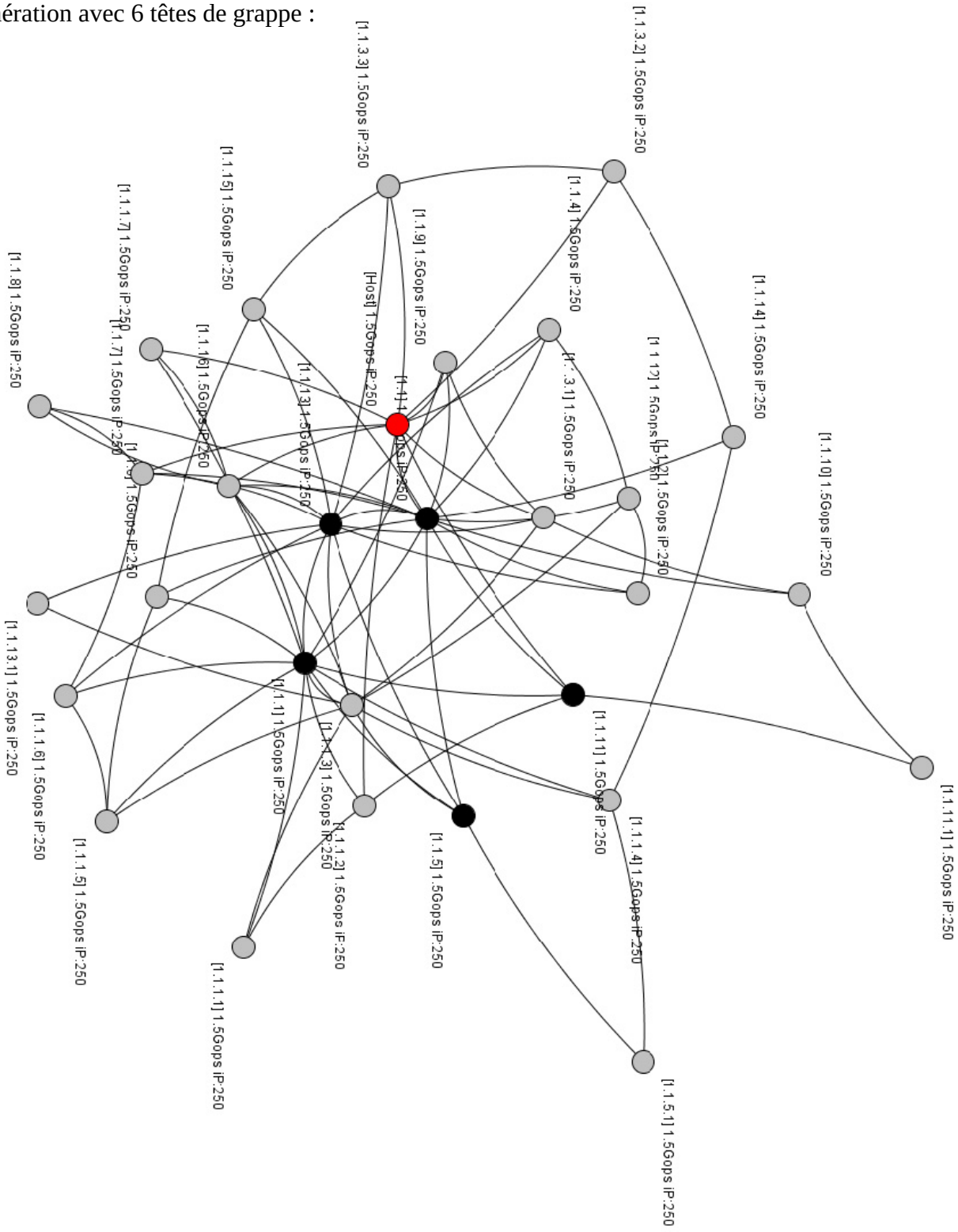
[24] F. Nimbona, S. Pierre, A. Quintero. "Clustering Concept and QoS Constraints in Dense Mobile Ad Hoc Networks", International Journal of Pervasive Computing and Communications, Special Issue on Wireless Networks and Pervasive Computing, Vol. 2, No. 2, pp. 61-67, Juin 2006.

[25] SUN Microsystems, JAVA : http://www.java.com (accédé en novembre 2009).

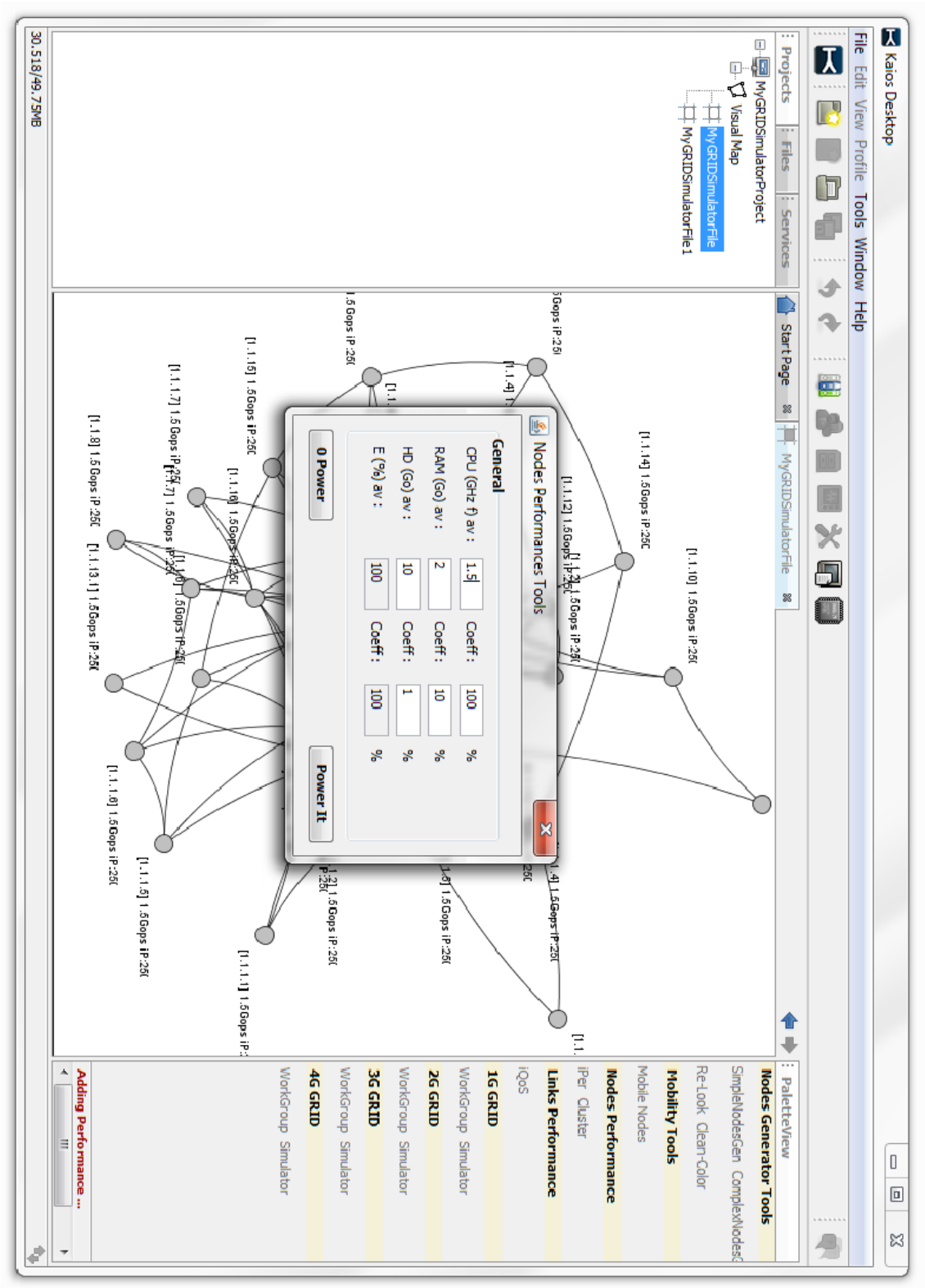
[26] Netbeans IDE : http://www.netbeans.net (accédé en novembre 2009).

ANNEXE A - **Interfaces du simulateur de GRID**

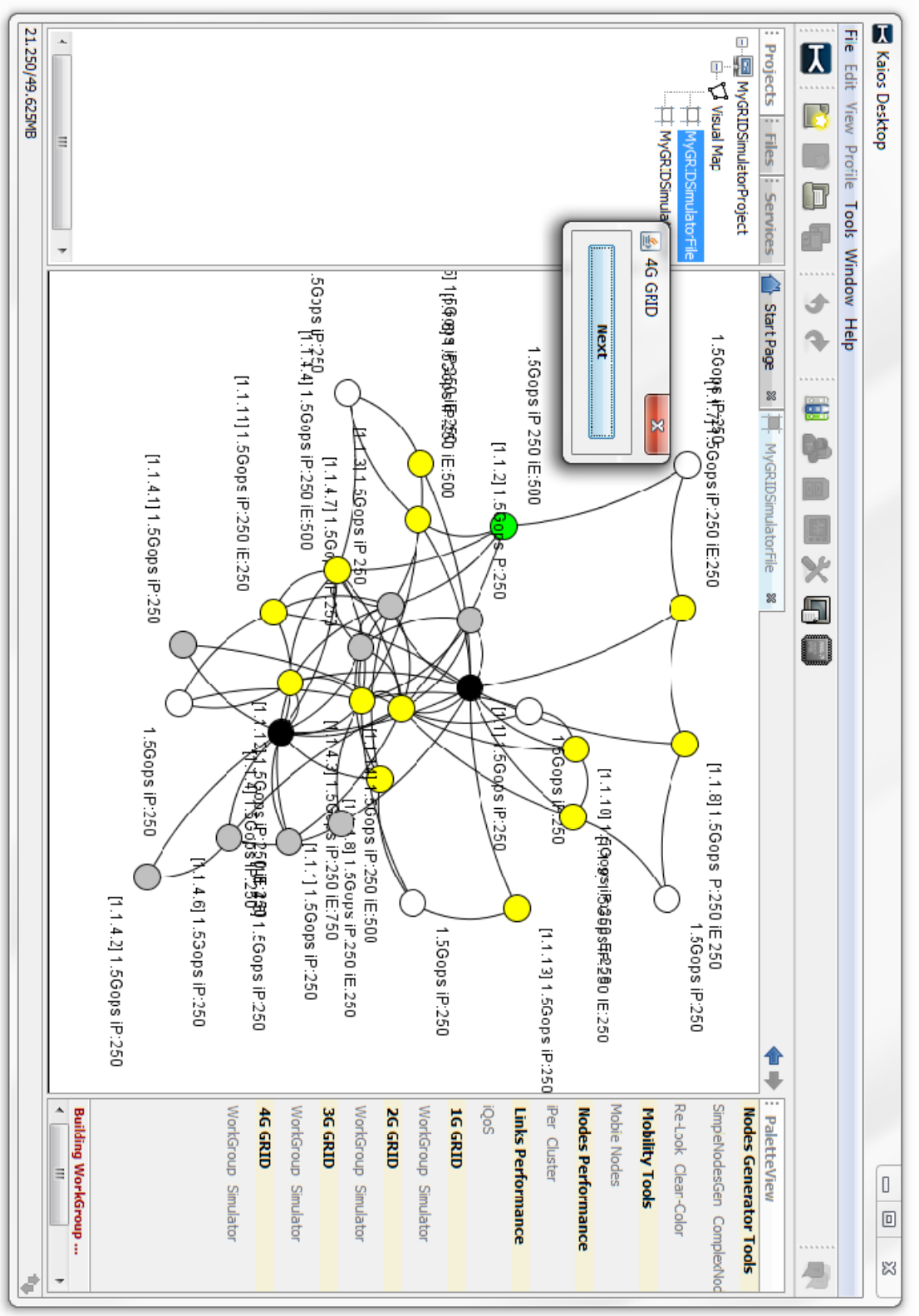
La figure suivante illustre un groupe de travail de 30 nœuds d'une GRID de nouvelle génération avec 6 têtes de grappe :



La figure suivante présente l'interface permettant de modifier les caractéristiques d'un nœud de la GRID :



La figure suivante présente l'interface permettant de suivre la formation d'un groupe de travail dans une GRID de nouvelle génération :



ANNEXE B - **Validation de l'interface principale du simulateur de GRID**

Pour chaque élément de l'interface principale présenté dans la section 4.1.7.1, nous énumérons les spécifications des fonctionnalités qu'il offre afin de valider le fonctionnement global de notre simulateur de GRID :

- Projets :
  - Ouvrir un projet.
  - Sélectionner un projet.
  - Ouvrir une simulation.
  - Sélectionner une simulation.

  1.6. Permettre de sauvegarder la configuration en cours.

  1.7. Permettre de charger une ancienne configuration.

- Réseau d'ordinateurs :
  - Sélectionner un nœud.
  - Déplacer un nœud.
  - Sélectionner un lien.

  1.5. Visualiser le réseau d'ordinateurs :

  1.5.1. Représenter les nœuds de différentes couleurs selon leurs rôles.

  1.5.2. Représenter les liens entre les nœuds.

  1.5.3. Afficher la somme de la puissance de calcul de toutes les ressources de la GRID.

- Palette d'outils de simulation :

  1. Permettre de former un réseau d'ordinateurs :

     1.1. Définir la génération de la GRID.

     1.2. Choisir le nombre de ressources dans la GRID.

     1.3. Déterminer les connexions entre les ressources.

     1.4. Déterminer le type d'environnement.

  2. Permettre d'éditer une ressource :

     2.1. Déterminer la puissance de calcul.

     2.2. Déterminer l'espace disque disponible.

2.3. Déterminer l’espace de la mémoire vive disponible.

2.4. Déterminer l’autonomie restante.

2.5. Visualiser l’indice de mobilité:

2.5.1. Déterminer le délai de temps moyen avant une déconnexion.

2.5.2. Déterminer la probabilité de reconnexion après une déconnexion.

2.6. Visualiser l’indice de performance:

2.6.1. Déterminer le coefficient de la puissance de calcul.

2.6.2. Déterminer le coefficient de l’espace disque.

2.6.3. Déterminer le coefficient de l’espace de la mémoire vive.

2.6.4. Déterminer le coefficient de l’autonomie restante.

2.7. Visualiser l’indice d’élection:

2.7.1. Déterminer le coefficient de l’indice de performance.

2.7.2. Déterminer le coefficient de l’indice de mobilité.

2.7.3. Déterminer le coefficient de l’indice de qualité de service du réseau.

3. Permettre d’éditer un lien entre deux ressources :

3.1. Visualiser l’indice de la qualité de service :

3.1.1. Déterminer le débit de chaque sens de la connexion.

3.1.2. Déterminer le délai de réponse de chaque sens de la connexion.

3.1.3. Déterminer le coefficient du poids du débit.

3.1.4. Déterminer le coefficient du poids du délai de réponse.

3.2. Déterminer la fiabilité du lien.

4. Permettre la formation de groupe de travail :

4.1. Pour la première génération de GRID :

4.1.1. Déterminer l’hôte.

4.1.2. Former une GRID avec les voisins directs de l’hôte.

4.1.3. Former un groupe de travail de la taille de la GRID.

4.2. Pour la deuxième génération de GRID :

4.2.1. Déterminer l'hôte.

4.2.2. Former une GRID avec les voisins directs de l'hôte.

4.2.3. Former un groupe de travail de la taille de la GRID.

4.3. Pour la troisième génération de GRID :

4.3.1. Déterminer l'hôte.

4.3.2. Déterminer la qualité de service de la requête.

4.3.3. Former un groupe de travail :

4.3.3.1. Ne considérer que les voisins directs de l'hôte.

4.3.3.2. Respecter l'indice de qualité de service de la requête.

4.4. Pour la GRID proposée de nouvelle génération :

4.4.1. Déterminer l'hôte.

4.4.2. Déterminer la qualité de service de la requête.

4.4.3. Former un groupe de travail :

4.4.3.1. Considérer toutes les ressources de la GRID.

4.4.3.2. Hiérarchiser le groupe

4.4.3.3. Respecter la qualité de service requis par la requête.

5. Pouvoir soumettre une requête à traiter au groupe de travail :

5.1. Pour la première génération de GRID :

5.1.1. Déterminer la taille de la requête à traiter.

5.1.2. Déterminer le délai entre chaque étape de la simulation.

5.1.3. Déterminer un temps d'arrêt d'une simulation.

5.2. Pour la deuxième génération de GRID :

5.2.1. Déterminer la taille de la requête à traiter.

5.2.2. Déterminer le délai entre chaque étape de la simulation.

5.2.3. Déterminer un temps d'arrêt d'une simulation.

5.3. Pour la troisième génération de GRID :

5.3.1. Déterminer la taille de la requête à traiter.

5.3.2. Déterminer le délai entre chaque étape de la simulation.

5.3.3. Déterminer un temps d'arrêt d'une simulation.

5.3.4. Visualiser la probabilité moyenne de reconnexion des nœuds.

5.4. Pour la GRID proposée de nouvelle génération :

5.4.1. Déterminer la taille de la requête à traiter.

5.4.2. Déterminer le délai entre chaque étape de la simulation.

5.4.3. Déterminer un temps d'arrêt d'une simulation.

5.4.4. Visualiser la probabilité moyenne de reconnexion des nœuds.

5.4.5. Visualiser la probabilité moyenne de déconnexion douce des nœuds.

- Log et résultats :

1.5.3. Afficher la somme de la puissance de calcul de toutes les ressources de la GRID.

5.5. Permettre d'exporter des données de simulation :

5.5.1. Exporter sous le format Excel le temps de simulation.

5.5.2. Exporter sous le format Excel l'avancement du traitement global de la requête.

- Barre d'outils générales :
  - Quitter le simulateur.

1.6. Permettre de sauvegarder la configuration en cours.

1.7. Permettre de charger une ancienne configuration.

ANNEXE C - **Performance de la nouvelle génération de GRID selon l'environnement**

Agrandissement des résultats présentés à la Figure 4.11 :